

3/5/1 (Item 1 from file: 351)
DIALOG(R) File 351:Derwent WPI
(c) 2004 Thomson Derwent. All rts. reserv.

015309997 **Image available**
WPI Acc No: 2003-370931/200335
XRPX Acc No: N03-295810

Multi-thread executing method in parallel processing system, involves
completing thread process by executing specific fork instruction based on
predetermined condition

Patent Assignee: NEC CORP (NIDE)
Inventor: MATSUSHITA S; OHSAWA T
Number of Countries: 003 Number of Patents: 003
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 20030014473	A1	20030116	US 2002189455	A	20020708	200335 B
JP 2003029985	A	20030131	JP 2001212247	A	20010712	200335
GB 2380573	A	20030409	GB 200216275	A	20020712	200335

Priority Applications (No Type Date): JP 2001212247 A 20010712
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 20030014473	A1		66	G06F-009/00	
JP 2003029985	A		38	G06F-009/46	
GB 2380573	A			G06F-009/38	

Abstract (Basic): US 20030014473 A1

NOVELTY - A specific fork instruction is selected for creating an
effective child thread during execution of parent thread. The
processing of thread is completed by executing the instruction upto an
address just before a starting address of the effective child thread.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is included for
parallel processing system.

USE - For performing multi-thread execution in parallel processing
system (claimed).

ADVANTAGE - The system is capable of assuring the Fork-Once
limitation at a time of program execution, thereby increasing the
probability of the fork with improved performance.

DESCRIPTION OF DRAWING(S) - The figure shows a flowchart explaining
the multi-thread executing process.

pp; 66 DwgNo 4/39

Title Terms: MULTI; THREAD; EXECUTE; METHOD; PARALLEL; PROCESS; SYSTEM;
COMPLETE; THREAD; PROCESS; EXECUTE; SPECIFIC; FORK; INSTRUCTION; BASED;
PREDETERMINED; CONDITION

Derwent Class: T01

International Patent Class (Main): G06F-009/00; G06F-009/38; G06F-009/46

International Patent Class (Additional): G06F-015/16; G06F-015/177

File Segment: EPI

3/5/2 (Item 1 from file: 347)
DIALOG(R) File 347:JAPIO
(c) 2004 JPO & JAPIO. All rts. reserv.

07536150 **Image available**
MULTITHREAD EXECUTION METHOD AND PARALLEL PROCESSOR SYSTEM

PUB. NO.: 2003-029985 A]
PUBLISHED: January 31, 2003 (20030131)
INVENTOR(s): OSAWA HIROSHI
MATSUSHITA SATOSHI
APPLICANT(s): NEC CORP
APPL. NO.: 2001-212247 [JP 2001212247]
FILED: July 12, 2001 (20010712)
INTL CLASS: G06F-009/46; G06F-015/16; G06F-015/177

ABSTRACT

PROBLEM TO BE SOLVED: To guarantee fork one time limitation at the time of executing a parallelized program.

SOLUTION: In the multithread execution method of dividing a single program into a plurality of threads and parallelly executing them in a plurality of processors, a program PE1 guarantees the fork one time limitation at the time of program execution by selecting one fork instruction generating a valid slave thread from a plurality of the fork instructions present in a master thread while executing the master thread by canceling (c, f) the slave thread in the case that the slave thread generated from the thread already exists or invalidating (d, g) all the fork instructions other than the fork instruction fork which succeeds in forking the slave thread first for each fork instruction fork of the thread during execution. The processor PE1 which generates the valid slave thread ends the processing of the thread by completing the execution to the instruction of an address immediately before the start address of the valid slave thread.

COPYRIGHT: (C)2003,JPO

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2003-29985

(P2003-29985A)

(43) 公開日 平成15年1月31日 (2003.1.31)

(51) Int.Cl.⁷

識別記号

F I

テーマコード(参考)

G 0 6 F 9/46
15/16
15/1773 6 0
6 1 0
6 7 4
6 8 1G 0 6 F 9/46
15/16
15/1773 6 0 B 5 B 0 4 5
6 1 0 Z 5 B 0 9 8
6 7 4 A
6 8 1 B

審査請求 有 請求項の数32 O L (全 38 頁)

(21) 出願番号 特願2001-212247(P2001-212247)

(22) 出願日 平成13年7月12日 (2001.7.12)

(71) 出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72) 発明者 大澤 拓

東京都港区芝五丁目7番1号 日本電気株式会社内

(72) 発明者 松下 智

東京都港区芝五丁目7番1号 日本電気株式会社内

(74) 代理人 100088959

弁理士 境 廣巳

Fターム(参考) 5B045 GG11

5B098 AA10 GA05 GC14

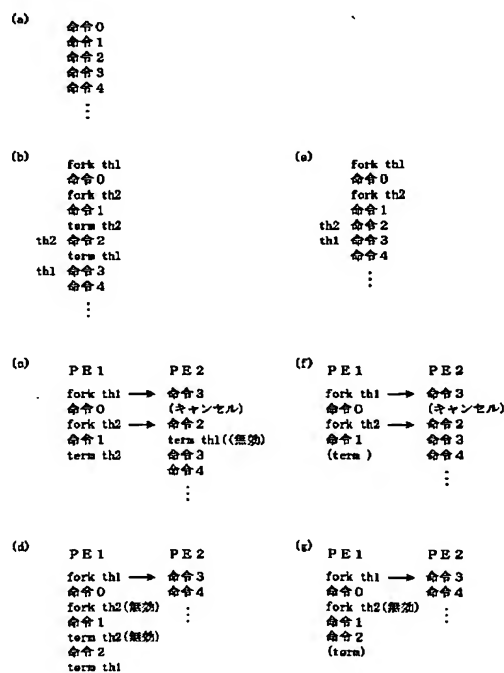
(54) 【発明の名称】 マルチスレッド実行方法及び並列プロセッサシステム

(57) 【要約】

【課題】 フォーク1回制限を並列化プログラムの実行時に保証する。

【解決手段】 単一のプログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法において、プログラムPE1は、実行中のスレッドのフォーク命令fork毎に、当該スレッドから生成された子スレッドが既に存在する場合にはその子スレッドをキャンセルするか(c, f)、最初に子スレッドのフォークに成功したフォーク命令fork以外の全てのフォーク命令を無効化することで(d, g)、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する1つのフォーク命令を親スレッド実行中に選択することによりフォーク1回制限をプログラム実行時に保証する。有効な子スレッドを生成したプロセッサPE1は、有効な子スレッドの開始アドレスの直前のアドレスの命令まで実行を完了することによりスレッドの処理を終了する。

【図1】



【特許請求の範囲】

【請求項 1】 単一のプログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法において、

各々の前記プロセッサは、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する 1 つのフォーク命令を親スレッド実行中に選択することによりフォーク 1 回制限をプログラム実行時に保証し、且つ、有効な子スレッドを生成したプロセッサは、有効な子スレッドの開始アドレスの直前のアドレスの命令まで実行を完了することによりスレッドの処理を終了することを特徴とするマルチスレッド実行方法。

【請求項 2】 親スレッドのフォーク命令毎に前記親スレッドから生成された子スレッドが既に存在する場合にはその子スレッドをキャンセルする請求項 1 記載のマルチスレッド実行方法。

【請求項 3】 親スレッドの実行を開始したプロセッサで最初に子スレッドのフォークに成功したフォーク命令以外の全てのフォーク命令を無効化する請求項 1 記載のマルチスレッド実行方法。

【請求項 4】 前記プロセッサは、各フォーク命令に対応して並列化プログラム中のフォーク先アドレスの直前に挿入されているターム命令のうち、有効な子スレッドをフォークしたフォーク命令に対応するターム命令によってスレッドの処理を終了する請求項 1 乃至 3 の何れか 1 項に記載のマルチスレッド実行方法。

【請求項 5】 前記プロセッサは、プログラムカウンタの値が有効な子スレッドの開始アドレスと一致したときにスレッドの処理を終了する請求項 1 乃至 3 の何れか 1 項に記載のマルチスレッド実行方法。

【請求項 6】 親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在しなかった場合、子スレッドの実行を開始できる他のプロセッサが生じるまで前記フォーク命令の実行をウエイトする請求項 1 乃至 3 の何れか 1 項に記載のマルチスレッド実行方法。

【請求項 7】 親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在しなかった場合、親スレッドのレジスタファイルの内容を退避させ、子スレッドの実行を開始できる他のプロセッサが生じた時点で前記退避した情報に基づいて子スレッドのフォークを行う請求項 1 乃至 3 の何れか 1 項に記載のマルチスレッド実行方法。

【請求項 8】 親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在しなかったフォーク命令は無効化する請求項 1 乃至 3 の何れか 1 項に記載のマルチスレッド実行方法。

【請求項 9】 親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在せず且つ親スレッドのレジスタファイルが更新さ

れる前に子スレッドの実行を開始できる他のプロセッサが生じなかったフォーク命令は無効化する請求項 1 乃至 3 の何れか 1 項に記載のマルチスレッド実行方法。

【請求項 10】 親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在せず且つ親スレッドのレジスタファイルのレジスタのうち子スレッドに継承すべきレジスタが更新される前に子スレッドの実行を開始できる他のプロセッサが生じなかったフォーク命令は無効化する請求項 1 乃至 3 の何れか 1 項に記載のマルチスレッド実行方法。

【請求項 1.1】 各々プログラムカウンタ及びレジスタファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行し且つスレッド実行のキャンセルが可能な仮実行用バッファを有する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジスタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親スレッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビットをリセットした状態で前記プロセッサに親スレッドの実行を開始させるステップ、(b) 親スレッド中の各フォーク命令の時点で前記プロセッサに設けたレジスタにフォーク先アドレスを保存し、前記フォークドビットがリセットされている場合は、前記プロセッサのフォーク先プロセッサがフリー状態のときは前記子スレッドをフォーク先プロセッサにフォークして前記フォークドビットをセットし、フォーク先プロセッサがビジー状態のときは前記フォーク命令を無効化し、前記フォークドビットがセットされている場合は、フォーク先プロセッサのスレッド実行をキャンセルして前記子スレッドをフォーク先プロセッサにフォークするステップ、(c) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 1.2】 各々プログラムカウンタ及びレジスタファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行し且つスレッド実行のキャンセルが可能な仮実行用バッファを有する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジスタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親ス

3

レッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビット及びフォーク有効ビットをリセットした状態で前記プロセッサに親スレッドの実行を開始させるステップ、(b) 親スレッド中の各フォーク命令の時点で前記プロセッサに設けたレジスタにフォーク先アドレスを保存すると共に前記フォーク有効ビットをセットするステップ、(c) 親スレッドのレジスタファイルが更新されたときに前記フォーク有効ビットをリセットするステップ、(d) 前記プロセッサの前記フォーク有効ビットがセットされている場合、前記フォークドビットがリセットされているときは前記プロセッサのフォーク先プロセッサがフリー状態になった時点で前記子スレッドをフォーク先プロセッサにフォークして前記フォークドビットをセットすると共に前記フォーク有効ビットをリセットし、前記フォークドビットがセットされているときはフォーク先プロセッサのスレッド実行をキャンセルして前記子スレッドをフォーク先プロセッサにフォークするステップ、(e) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 13】 各々プログラムカウンタ及びレジスタファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行し且つスレッド実行のキャンセルが可能な仮実行用バッファを有する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジスタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親スレッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビット及びフォーク有効ビットをリセットした状態で前記プロセッサに親スレッドの実行を開始させるステップ、(b) 親スレッド中の各フォーク命令の時点で前記プロセッサに設けたレジスタにフォーク先アドレスを保存すると共に前記フォーク有効ビットをセットするステップ、(c) 親スレッドのレジスタファイルのレジスタの内、子ステップに継承すべきレジスタが更新されたときに前記フォーク有効ビットをリセットするステップ、(d) 前記プロセッサの前記フォーク有効ビットがセットされている場合、前記フォークドビットがリセットされているときは前記プロセッサのフォーク先プロセッサがフリー状態になった時点で

4

前記子スレッドをフォーク先プロセッサにフォークして前記フォークドビットをセットすると共に前記フォーク有効ビットをリセットし、前記フォークドビットがセットされているときはフォーク先プロセッサのスレッド実行をキャンセルして前記子スレッドをフォーク先プロセッサにフォークするステップ、(e) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 14】 各々プログラムカウンタ及びレジスタファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行し且つスレッド実行のキャンセルが可能な仮実行用バッファを有する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジスタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親スレッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビットをリセットした状態で前記プロセッサに親スレッドの実行を開始させるステップ、(b) 親スレッド中の各フォーク命令の時点で前記プロセッサに設けたレジスタにフォーク先アドレスを保存し、前記フォークドビットがリセットされている場合は、前記プロセッサのフォーク先プロセッサがフリー状態であるときは直ちに、フリー状態でないときはフリー状態になるまで前記フォーク命令をウエイトしてから前記子スレッドをフォーク先プロセッサにフォークして前記フォークドビットをセットし、前記フォークドビットがセットされている場合は、フォーク先プロセッサのスレッド実行をキャンセルして前記子スレッドをフォーク先プロセッサにフォークするステップ、(c) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 15】 各々プログラムカウンタ及びレジスタファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行し且つスレッド実行のキャンセルが可能な仮実行用バッファを有する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジ

5

スタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親スレッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビットをリセットした状態で前記プロセッサに親スレッドの実行を開始させるステップ、(b) 親スレッド中の各フォーク命令の時点で前記プロセッサに設けたレジスタにフォーク先アドレスを保存し、前記フォークドビットがリセットされている場合は、前記プロセッサのフォーク先プロセッサがフリー状態のときは直ちに前記レジスタファイルの内容に基づいて前記子スレッドをフォーク先プロセッサにフォークして前記フォークドビットをセットし、前記プロセッサのフォーク先プロセッサがビジー状態のときは前記レジスタファイルの内容を前記プロセッサに設けた退避バッファに退避してフォーク先プロセッサがフリー状態になるまで子スレッドのフォークを保留し、前記フォークドビットがセットされている場合は、フォーク先プロセッサのスレッド実行をキャンセルして前記子スレッドをフォーク先プロセッサにフォークするステップ、(c) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 16】 請求項 11 乃至 15 の何れか 1 項に記載のマルチスレッド実行方法において、前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップの代わりに、前記プロセッサは、各フォーク命令に対応して並列化プログラム中のフォーク先アドレスの直前に挿入されているターム命令のうち、有効な子スレッドをフォークしたフォーク命令に対応するターム命令によってスレッドの処理を終了するステップを含むマルチスレッド実行方法。

【請求項 17】 各々プログラムカウンタ及びレジスタファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジスタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親スレッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビットをリセットした状態で前記プ

6

ロセッサに親スレッドの実行を開始させるステップ、

(b) 親スレッド中のフォーク命令の時点で、前記フォークドビットがリセットされている場合は、前記親スレッドを実行しているプロセッサのフォーク先プロセッサがフリー状態のときは前記子スレッドをフォークして前記フォークドビットをセットすると共に前記プロセッサに設けたレジスタにフォーク先アドレスを保存し、フォーク先プロセッサがビジー状態のときは前記フォーク命令を無効化し、前記フォークドビットがセットされている場合は、前記フォーク命令を無効化するステップ、

(c) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 18】 各々プログラムカウンタ及びレジスタファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジスタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親スレッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビットをリセットした状態で前記プロセッサに親スレッドの実行を開始させるステップ、

(b) 親スレッド中のフォーク命令の時点で、前記フォークドビットがリセットされている場合は、前記プロセッサに設けたフォーク有効ビットをセットすると共に前記プロセッサに設けたレジスタにフォーク先アドレスを保存し、前記フォークドビットがセットされている場合は、前記フォーク命令を無効化するステップ、(c) 親スレッドのレジスタファイルが更新されたときに前記フォーク有効ビットをリセットするステップ、(d) 前記プロセッサの前記フォーク有効ビットがセットされている場合、前記親スレッドを実行しているプロセッサのフォーク先プロセッサがフリー状態であれば前記プロセッサの前記レジスタに保存されたフォーク先アドレスから始まる子スレッドをフォーク先プロセッサにフォークして前記プロセッサの前記フォークドビットをセットし且つ前記フォーク有効ビットをリセットするステップ、

(e) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 19】 各々プログラムカウンタ及びレジスタ

ファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジスタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親スレッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビットをリセットした状態で前記プロセッサに親スレッドの実行を開始させるステップ、

(b) 親スレッド中のフォーク命令の時点で、前記フォークドビットがリセットされている場合は前記プロセッサに設けたフォーク有効ビットをセットすると共に前記プロセッサに設けたレジスタにフォーク先アドレスを保存し、前記フォークドビットがセットされている場合は前記フォーク命令を無効化するステップ、(c) 親スレッドのレジスタファイルのレジスタのうち子スレッドに継承すべきレジスタが更新されたときに前記フォーク有効ビットをリセットするステップ、(d) 前記フォーク有効ビットがセットされている場合、前記親スレッドを実行しているプロセッサのフォーク先プロセッサがフリー状態であれば前記子スレッドをフォーク先プロセッサにフォークして前記フォークドビットをセットし且つ前記フォーク有効ビットをリセットするステップ、(e) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 20】 各々プログラムカウンタ及びレジスタファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジスタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親スレッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビットをリセットした状態で前記プロセッサに親スレッドの実行を開始させるステップ、

(b) 親スレッド中のフォーク命令の時点で、前記フォークドビットがリセットされている場合は、前記親スレッドを実行しているプロセッサのフォーク先プロセッサがフリー状態のときは直ちに、フリー状態でなければフォ

ーク先プロセッサがフリー状態になるまで前記フォーク命令をウエイトしてから前記子スレッドをフォークして前記フォークドビットをセットすると共に前記プロセッサに設けたレジスタにフォーク先アドレスを保存し、前記フォークドビットがセットされている場合は、前記フォーク命令を無効化するステップ、(c) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 21】 各々プログラムカウンタ及びレジスタファイルを独立に有し前記プログラムカウンタに従ってスレッドの命令を同時にフェッチ、解釈、実行する複数のプロセッサを備え、何れかの前記プロセッサで実行されている親スレッド中のフォーク命令によって指定されたフォーク先アドレスから始まる子スレッドの実行を、前記フォーク命令時点の前記親スレッドのレジスタファイルのうち少なくとも前記子スレッドに必要なレジスタの値を前記子スレッドに継承させて、前記親スレッドを実行しているプロセッサのフォーク先プロセッサに開始させる機能を備えた並列プロセッサシステムにおけるマルチスレッド実行方法において、(a) 前記プロセッサに設けたフォークドビットをリセットした状態で前記プロセッサに親スレッドの実行を開始させるステップ、

(b) 親スレッド中のフォーク命令の時点で、前記フォークドビットがリセットされている場合は、前記親スレッドを実行しているプロセッサのフォーク先プロセッサがフリー状態のときは直ちに前記レジスタファイルの内容に基づいて前記子スレッドを隣接プロセッサにフォークして前記フォークドビットをセットし、フォーク先プロセッサがビジー状態のときは前記レジスタファイルの内容を前記プロセッサに設けた退避バッファに退避してフォーク先プロセッサがフリー状態になるまで子スレッドのフォークを保留し、前記フォークドビットがセットされている場合は、前記フォーク命令を無効化するステップ、(c) 前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップ、を含むことを特徴とするマルチスレッド実行方法。

【請求項 22】 請求項 17 乃至 21 の何れか 1 項に記載のマルチスレッド実行方法において、前記フォークドビットがセットされており且つプログラムカウンタの値が前記レジスタに保存された前記フォーク先アドレスと一致した前記プロセッサはスレッドの処理を終了するステップの代わりに、前記プロセッサは、各フォーク命令に対応して並列化プログラム中のフォーク先アドレスの直前に挿入されているターム命令のうち、有効な子スレッドをフォークしたフォーク命令に対応するターム命令

によってスレッドの処理を終了するステップを含むマルチスレッド実行方法。

【請求項 23】 単一のプログラムを複数のスレッドに分割し複数のプロセッサで並列に実行する並列プロセッサシステムにおいて、各々の前記プロセッサは、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する 1 つのフォーク命令を親スレッド実行中に選択することによりフォーク 1 回制限をプログラム実行時に保証する手段と、有効な子スレッドを生成したプロセッサは、有効な子スレッドの開始アドレスの直前のアドレスの命令まで実行を完了することによりスレッドの処理を終了する手段とを備えたことを特徴とする並列プロセッサシステム。

【請求項 24】 親スレッドのフォーク命令毎に前記親スレッドから生成された子スレッドが既に存在する場合にはその子スレッドをキャンセルする請求項 23 記載の並列プロセッサシステム。

【請求項 25】 親スレッドの実行を開始したプロセッサで最初に子スレッドのフォークに成功したフォーク命令以外の全てのフォーク命令を無効化する請求項 23 記載の並列プロセッサシステム。

【請求項 26】 前記プロセッサは、各フォーク命令に対応して並列化プログラム中のフォーク先アドレスの直前に挿入されているターム命令のうち、有効な子スレッドをフォークしたフォーク命令に対応するターム命令によってスレッドの処理を終了する請求項 23 乃至 25 の何れか 1 項に記載の並列プロセッサシステム。

【請求項 27】 前記プロセッサは、プログラムカウンタの値が有効な子スレッドの開始アドレスと一致したときにスレッドの処理を終了する請求項 23 乃至 25 の何れか 1 項に記載の並列プロセッサシステム。

【請求項 28】 親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在しなかった場合、子スレッドの実行を開始できる他のプロセッサが生じるまで前記フォーク命令の実行をウエイトする請求項 23 乃至 25 の何れか 1 項に記載の並列プロセッサシステム。

【請求項 29】 親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在しなかった場合、親スレッドのレジスタファイルの内容を退避させ、子スレッドの実行を開始できる他のプロセッサが生じた時点で前記退避した情報に基づいて子スレッドのフォークを行う請求項 23 乃至 25 の何れか 1 項に記載の並列プロセッサシステム。

【請求項 30】 親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在しなかったフォーク命令は無効化する請求項 23 乃至 25 の何れか 1 項に記載の並列プロセッサシステム。

【請求項 31】 親スレッドの実行開始後、フォーク命

令の時点で子スレッドの実行を開始できる他のプロセッサが存在せず且つ親スレッドのレジスタファイルが更新される前に子スレッドの実行を開始できる他のプロセッサが生じなかったフォーク命令は無効化する請求項 23 乃至 25 の何れか 1 項に記載の並列プロセッサシステム。

【請求項 32】 親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在せず且つ親スレッドのレジスタファイルのレジスタのうち子スレッドに継承すべきレジスタが更新される前に子スレッドの実行を開始できる他のプロセッサが生じなかったフォーク命令は無効化する請求項 23 乃至 25 の何れか 1 項に記載の並列プロセッサシステム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は並列プロセッサシステムにおけるプログラム並列実行方法に関し、より具体的には単一のプログラムを複数のスレッドに分割して複数のプロセッサにより並列に実行するマルチスレッド実行方法及び並列プロセッサシステムに関する。

【0002】

【従来の技術】単一のプログラムを並列プロセッサシステムで並列に処理する手法として、プログラムをスレッドと呼ぶ命令流に分割して複数のプロセッサで並列に実行するマルチスレッド実行方法があり、この方法を記載した文献として、特開平 10-27108 号公報（以下、文献 1 と称す）、「On Chip Multiprocessor 指向 制御並列アーキテクチャ MUSCAT の提案」（並列処理シンポジウム JSP97 論文集、情報処理学会、pp. 229-236、May 1997）（以下、文献 2 と称す）、特開平 10-78880 号公報（以下、文献 3 と称す）等がある。以下、これらの従来文献に記載されたマルチスレッド実行方法について説明する。

【0003】一般にマルチスレッド実行方法において、他のプロセッサ上に新たなスレッドを生成することを、スレッドをフォーク（fork）すると言い、フォーク動作を行った側のスレッドを親スレッド、生成された新しいスレッドを子スレッド、スレッドをフォークする箇所をフォーク点、子スレッドの先頭箇所をフォーク先アドレスまたは子スレッドの開始点と呼ぶ。文献 1～3 では、スレッドのフォークを指示するためにフォーク点にフォーク命令が挿入される。フォーク命令にはフォーク先アドレスが指定され、フォーク命令の実行によりそのフォーク先アドレスから始まる子スレッドが他プロセッサ上に生成され、子スレッドの実行が開始される。また、スレッドの処理を終了させるターム（term）命令と呼ばれる命令が用意されており、各プロセッサはターム命令を実行することによりスレッドの処理を終了する。

【0004】図37に従来のマルチスレッド実行方法の処理の概要を示す。同図(a)は3つのスレッドA、B、Cに分割された単一のプログラムを示す。このプログラムを単一のプロセッサで処理する場合、同図(b)に示すように1つのプロセッサPEがスレッドA、B、Cを順番に処理していく。これに対して文献1~3のマルチスレッド実行方法では、同図(c)に示すように、1つのプロセッサPE1にスレッドAを実行させ、プロセッサPE1でスレッドAを実行している最中に、スレッドAに埋め込まれたフォーク命令によってスレッドBを他のプロセッサPE2に生成し、プロセッサPE2においてスレッドBを実行させる。また、プロセッサPE2はスレッドBに埋め込まれたフォーク命令によってスレッドCをプロセッサPE3に生成する。プロセッサPE1、PE2はそれぞれスレッドB、Cの開始点の直前に埋め込まれたターム命令によってスレッドの処理を終了し、プロセッサPE3はスレッドCの最後の命令を実行すると、その次の命令(一般にはシステムコール命令)を実行する。このように複数のプロセッサでスレッドを同時に並行して実行することにより、逐次処理に比べて性能の向上が図られる。

【0005】従来の他のマルチスレッド実行方法として、図37(d)に示すように、スレッドAを実行しているプロセッサPE1からフォークを複数回行うことにより、プロセッサPE2にスレッドBを、またプロセッサPE3にスレッドCをそれぞれ生成するマルチスレッド実行方法も存在する。この図37(d)のモデルに対して、同図(c)に示したようにスレッドはその生存中に高々1回に限って有効な子スレッドを生成することができるという制約を課したマルチスレッド実行方法をフォーク1回モデルと呼ぶ。フォーク1回モデルでは、スレッド管理の大幅な簡略化が可能となり、現実的なハードウェア規模でスレッド管理部のハードウェア化が実現できる。また、個々のプロセッサは子スレッドを生成する他プロセッサが1プロセッサに限定されるため、隣接するプロセッサを単方向にリング状に接続した並列プロセッサシステムでマルチスレッド実行が可能となる。本発明はこのようなフォーク1回モデルを前提とする。

【0006】ここで、フォーク命令時、子スレッドを生成できる空きのプロセッサが存在しない場合、従来は次の2通りの方法の何れかを採用している。

(1) 親スレッドを実行しているプロセッサは、子スレッドを生成できる空きのプロセッサが生じるまで、フォーク命令の実行をウェイトする。

(2) 親スレッドを実行しているプロセッサは、フォーク先アドレス及びフォーク点におけるレジスタファイルの内容を裏面の物理レジスタに保存して親スレッドの後続処理を続行する。裏面の物理レジスタに保存されたフォーク先アドレス及びレジスタファイルの内容は、子スレッドを生成できる空きのプロセッサが生じた時点で参

照され、子スレッドが生成される。

【0007】親スレッドが子スレッドを生成し、子スレッドに所定の処理を行わせるには、親スレッドのフォーク点におけるレジスタファイル中のレジスタのうち少なくとも子スレッドに必要なレジスタの値を親スレッドから子スレッドに引き渡す必要がある。このスレッド間のデータ引き渡しコストを削減するために、文献2及び3では、スレッド生成時のレジスタ値継承機構をハードウェア的に備えている。これは、スレッド生成時に親スレッドのレジスタファイルの内容を子スレッドに全てコピーするものである。子スレッド生成後は、親スレッドと子スレッドのレジスタ値の変更は独立となり、レジスタを用いたスレッド間のデータの引き渡しは行われない。スレッド間のデータ引き渡しに関する他の従来技術としては、レジスタの値を命令によりレジスタ単位で個別に転送する機構を備えた並列プロセッサシステムも提案されている。

【0008】マルチスレッド実行方法では、実行の確定した先行スレッドを並列に実行することを基本とするが、実際のプログラムでは実行の確定するスレッドが十分に得られない場合も多い。また、動的に決定される依存やコンパイラ解析能力の限界等により並列化率が低く抑えられ所望の性能が得られない可能性が生じる。このため文献1では、制御投機を導入し、ハードウェア的にスレッドの投機実行をサポートしている。制御投機では、実行する可能性の高いスレッドを実行確定前に投機的に実行する。投機状態のスレッドは、実行の取り消しがハードウェア上可能である範囲内で仮実行を行う。子スレッドが仮実行を行っている状態を仮実行状態と言い、子スレッドが仮実行状態にあるとき親スレッドはスレッド仮生成状態にあると言う。仮実行状態の子スレッドでは共有メモリ及びキャッシュメモリへの書き込みは抑制され、別途設けた仮実行用バッファ(temporary buffer)に対して書き込みが行われる。投機が正しいことが確定すると、親スレッドから子スレッドに対して投機成功通知が出され、子スレッドは仮実行用バッファの内容を共有メモリ及びキャッシュメモリに反映し、仮実行用バッファを用いない通常の状態となる。また親スレッドはスレッド仮生成状態からスレッド生成状態となる。他方、投機が失敗したことが確定すると、親スレッドでスレッド破棄命令(abort)が実行され、子スレッド以下の実行がキャンセルされる。また、親スレッドはスレッド仮生成状態からスレッド未生成状態となり、再び子スレッドの生成が可能にある。つまり、フォーク1回モデルではスレッド生成は高々1回に限定されるが、制御投機を行い、投機が失敗した場合には再びフォークが可能となる。この場合においても、有効な子スレッドは高々1つである。

【0009】その他、文献2に記載のMUSCATでは、スレッド間の同期命令など、スレッドの並列動作を

柔軟に制御するための専用命令が数多く用意されている。

【0010】

【発明が解決しようとする課題】スレッドはその生存中に高々1回に限って有効な子スレッドを生成するというフォーク1回モデルのマルチスレッド実行を実現するために、従来は文献2等に表示されるように、逐次処理プログラムから並列化プログラムを生成するコンパイルの段階で、全てのスレッドが有効なフォークを1回しか実行しない命令コードになるように制限していた。即ち、フォーク1回制限を並列化プログラム上において静的に保証していた。

【0011】しかし、分割コンパイル、関数呼び出しなどの問題により、コンパイラがフォーク1回制限を守ることは難しい。従来のマルチスレッド実行方法及び並列プロセッサシステムでは、フォーク1回制限が守られていない並列化プログラムは正しく実行できない。例えば図38に示すようなmain関数及びfunc関数を含むプログラムにおいて、同図(a)に示すようにmain関数及びfunc関数の双方にフォーク命令が挿入されていると、ブロックaからブロックbに分岐する制御フローが実行される場合にはフォーク1回制限は守られるが、ブロックaからブロックcに分岐する制御フローが実行されると、同じスレッドからフォークが2回行われるため、フォーク1回制限が保証されず正常な実行が行えない。このため、従来はコンパイルの段階でmain関数かfunc関数の何れか一方にのみフォーク命令を挿入することで、フォーク1回制限を保証する必要があった。図38(b)に、func関数にのみフォーク命令を挿入し、main関数におけるブロックaの時点におけるブロックdの先行実行を断念した並列化プログラムの例を示す。

【0012】本発明はこのような事情に鑑みて提案されたものであり、その目的は、フォーク1回制限の保証のない並列化プログラムであってもフォーク1回モデルによるマルチスレッド実行が行える新規なマルチスレッド実行方法及び並列プロセッサシステムを提供することにある。

【0013】

【課題を解決するための手段】本発明は、単一のプログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法において、各々の前記プロセッサは、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する1つのフォーク命令を親スレッド実行中に選択することによりフォーク1回制限をプログラム実行時に保証し、且つ、有効な子スレッドを生成したプロセッサは、有効な子スレッドの開始アドレスの直前のアドレスの命令まで実行を完了することによりスレッドの処理を終了することを基本とする。

【0014】より具体的には、第1の発明は、親スレッドのフォーク命令毎に当該親スレッドから生成された子スレッドが既に存在する場合にはその子スレッドをキャンセルすることで、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する1つのフォーク命令を親スレッド実行中に選択し、フォーク1回制限をプログラム実行時に保証する。

【0015】また、第2の発明は、親スレッドの実行を開始したプロセッサで最初に子スレッドのフォークに成功したフォーク命令以外の全てのフォーク命令を無効化することで、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する1つのフォーク命令を親スレッド実行中に選択し、フォーク1回制限をプログラム実行時に保証する。

【0016】有効な子スレッドを生成したプロセッサに、有効な子スレッドの開始アドレスの直前のアドレスの命令まで実行を完了することによりスレッドの処理を終了させるには、プロセッサが各フォーク命令に対応して並列化プログラム中のフォーク先アドレスの直前に挿入されているターム命令のうち、有効な子スレッドをフォークしたフォーク命令に対応するターム命令によってスレッドの処理を終了するように構成しても良いし、各プロセッサのプログラムカウンタの値が有効な子スレッドの開始アドレスと一致したときにスレッドの処理を終了するように構成しても良い。

【0017】また、親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在しなかった場合、文献1に記載されるように、子スレッドの実行を開始できる他のプロセッサが生じるまでフォーク命令の実行をウエイトしたり、レジスタファイルの内容を退避させて子スレッドの実行を開始できる他のプロセッサが生じた時点で前記退避した情報に基づいて子スレッドのフォークを行うようにしても良い。更に、親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在しなかったフォーク命令は無効化したり、親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在せず且つ親スレッドのレジスタファイルが更新される前に子スレッドの実行を開始できる他のプロセッサが生じなかったフォーク命令は無効化したり、親スレッドの実行開始後、フォーク命令の時点で子スレッドの実行を開始できる他のプロセッサが存在せず且つ親スレッドのレジスタファイルのレジスタのうち子スレッドに継承すべきレジスタが更新される前に子スレッドの実行を開始できる他のプロセッサが生じなかったフォーク命令は無効化したりする方法を採用することも可能である。

【0018】

【作用】本発明にあつては、各プロセッサは、親スレッドのフォーク命令毎に当該親スレッドから生成された子

スレッドが既に存在する場合にはその子スレッドをキャンセルし（第1の発明）、親スレッドの実行を開始したプロセッサで最初に子スレッドのフォークに成功したフォーク命令以外の全てのフォーク命令を無効化することで（第2の発明）、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する1つのフォーク命令を親スレッド実行中に選択するため、フォーク1回制限の保証のない並列化プログラムであってもフォーク1回制限をプログラム実行時に保証することができ、また、有効な子スレッドを生成したプロセッサは、有効な子スレッドの開始アドレスの直前のアドレスの命令まで実行を完了することによりスレッドの処理を終了することにより、プログラムの処理を支障なく遂行できる。

【0019】次に本発明の作用を、その理解を容易にするために簡略化したプログラム例に基づいて説明する。

図1(a)は逐次処理プログラムの例を示し、同図(b)はこの逐次処理プログラムから生成された並列化プログラムの例を示す。この例の並列化プログラム中、`fork thi`は、アドレス`thi`以降の命令を子スレッドとしてフォークすることを指示するフォーク命令を示し、`term thi`は`fork thi`に対応するターム命令を示す。同図(b)の並列化プログラムでは、1つのスレッドから複数回フォークを行うように記述されており、フォーク1回制限が保証されていない。

【0020】図1(c)は同図(b)の並列化プログラムを第1の発明で実行したシーケンスを示す。プロセッサPE1はフォーク命令`fork th1`によりプロセッサPE2に子スレッドをフォークし、プロセッサPE2は命令3から子スレッドの実行を開始する。プロセッサPE1は引き続き命令0を実行し、再びフォーク命令`fork th2`が現れたので、プロセッサPE2に子スレッドをフォークする。このときプロセッサPE2上の実行中のスレッドはキャンセルされ、プロセッサPE2は命令2から新たな子スレッドの実行を開始する。プロセッサPE1は引き続き命令1を実行し、次の命令は有効な子スレッドをフォークしたフォーク命令`fork th2`に対応するターム命令`term th2`なので、スレッドの処理を終了する。他方、プロセッサPE2は、実行中のスレッドから未だ有効な子スレッドをフォークしていないので、命令2の次のターム命令`term th1`は無効化し、命令3、命令4と実行を進める。プロセッサPE1で実行されるスレッドからは有効なスレッドは高々1回しかフォークしていないため、フォーク1回制限がプログラム実行時に保証されている。また、最終的に、プロセッサPE1では命令0、命令1がその順に、プロセッサPE2では命令2、命令3、命令4がその順にそれぞれ実行されており、プログラムの処理を支障なく遂行できる。

【0021】図1(d)は同図(b)の並列化プログラ

ムを第2の発明で実行したシーケンスを示す。プロセッサPE1はフォーク命令`fork th1`によりプロセッサPE2に子スレッドをフォークし、プロセッサPE2は命令3から子スレッドの実行を開始する。プロセッサPE1は引き続き命令0を実行し、再びフォーク命令`fork th2`が現れるが既に子スレッドを1回フォークしているのでそれを無効化し、次の命令1を実行する。更に、プロセッサPE1はターム命令`term th2`は有効な子スレッドをフォークしたフォーク命令`fork th1`に対応するターム命令でないので無効化し、引き続き命令2を実行し、有効な子スレッドをフォークしたフォーク命令`fork th1`に対応するターム命令`term th1`を実行した時点で、スレッドの処理を終了する。プロセッサPE1で実行されるスレッドからは有効なスレッドは高々1回しかフォークしていないため、フォーク1回制限がプログラム実行時に保証されている。また、最終的に、プロセッサPE1では命令0、命令1、命令2がその順に、またプロセッサPE2では命令3、命令4がその順にそれぞれ実行されており、プログラムの処理を支障なく遂行できる。

【0022】図1(e)は図1(b)の並列化プログラム中からターム命令を取り除いた並列化プログラムを示し、図1(f)は同図(e)の並列化プログラムを第1の発明で実行したシーケンスを、図1(g)は同図

(e)の並列化プログラムを第2の発明で実行したシーケンスをそれぞれ示す。ターム命令を使わない場合、各プロセッサはプログラムカウンタの値が有効な子スレッドの開始アドレスと一致したときにスレッドの処理を終了する。図1(f)では、有効な子スレッドはフォーク命令`fork th2`でフォークされたスレッドであり、その開始アドレスは命令2なので、プロセッサPE1はプログラムカウンタの値が命令2のアドレスと一致すると終了している。また、図1(g)では、有効な子スレッドはフォーク命令`fork th1`でフォークされたスレッドであり、その開始アドレスは命令3なので、プロセッサPE1はプログラムカウンタの値が命令3のアドレスと一致すると終了している。

【0023】図1では、フォークが入れ子になっているプログラム例を用いて本発明の作用を説明したが、フォークが入れ子になっていないプログラムであっても同様にフォーク1回制限を実行時に保証し且つプログラムの処理を支障なく遂行することが可能である。

【0024】なお、本発明においては、親スレッドから子スレッドへのフォーク時におけるレジスタの値の継承は、フォーク命令時点の親スレッドのレジスタファイルのうち少なくとも子スレッドで必要なレジスタだけを対象とすれば足りる。このための具体的なレジスタ継承機構としては、文献2及び文献3に記載されるようにスレッド生成時に親スレッドのレジスタファイルの内容すべてを子スレッドのレジスタファイルにコピーするもので

10

20

30

40

50

あっても良いし、レジスタ転送量の削減を図るために必要なレジスタの値だけを命令によりレジスタ単位で個別に転送するものであっても良い。

【0025】

【発明の実施の形態】次に第1の発明の実施例について図面を参照して詳細に説明する。

【0026】

【第1の発明の第1の実施例】図2を参照すると、本発明の並列プロセッサシステムの一例は、4スレッド並列実行型プロセッサであり、4個のプロセッサ1-i (i = 0~3) が信号線2-iによってスレッド管理部3に接続されると共に、信号線4-iによって共有のメモリ5に接続されている。また、プロセッサ1-i相互間は通信バス6で接続されている。この例では、4スレッド並列実行型プロセッサを取り上げたが、8スレッドや16スレッドの並列実行型プロセッサ等、一般にn (≥ 2) スレッド並列実行型プロセッサに対して本発明は適用可能である。全てのプロセッサ1-i、メモリ5及びスレッド管理部3はクロックに同期して動作する。また、好ましくは、全てのプロセッサ1-iはメモリ5及びスレッド管理部3と共に1つの半導体チップ上に集積化される。

【0027】各プロセッサ1-iは、プログラムカウンタ(以下、PCと称す)及びレジスタファイルを独立に有し、PCに従って、メモリ5中のスレッドの命令を同時にフェッチ、解釈、実行する機能を有している。また、各プロセッサ1-iは、スレッド実行の取り消し(キャンセル)が可能のように仮実行用バッファ(temporary buffer)を有している。各プロセッサ1-iは、スレッド管理部3から信号線2-iを通じてターゲットPC値を伴うスレッド開始要求7cが送信された時点で、仮実行用バッファを使ってスレッドの実行を仮実行状態で開始する。この時点で当該プロセッサ1-iはビジー状態として管理される。スレッドの実行を終了するプロセッサ1-iは、スレッド管理部3に対して信号線2-iを通じてスレッド終了通知7dを送信する。このスレッド終了通知7dがスレッド管理部3で受理された時点で、当該プロセッサ1-iはフリー状態として管理され、プロセッサ1-iにスレッド終了許可7eが返却される。プロセッサ1-iはスレッド終了許可7eを受信した時点で仮実行状態を解き、仮実行用バッファの内容を共有のメモリ5及び図示しないキャッシュメモリに反映させ、スレッドの実行を終える。

【0028】各プロセッサ1-iは、実行中の親スレッドに存在するフォーク命令によって他のプロセッサ1-j (i ≠ j) に子スレッドをフォークすることができる。本実施例では、文献1~3に記載されるように、スレッド管理の簡便化のためにプロセッサ1-iから子スレッドをフォークできるプロセッサを、プロセッサ1-iの一方の隣接プロセッサ(プロセッサ1-0はプロセ

ッサ1-1、プロセッサ1-1はプロセッサ1-2、プロセッサ1-2はプロセッサ1-3、プロセッサ1-3はプロセッサ1-0)に限定している。このようなモデルを、以下、リング型フォークモデルと称す。

【0029】各プロセッサ1-iは、子スレッドのフォークを行う際、信号線2-iを通じてスレッド管理部3に対し、子スレッドのフォーク先アドレス(開始PC値)及び既に子スレッドを生成したことがあるか否かを示す信号(子スレッド生成済信号)を伴うフォーク要求7aを送信する。スレッド管理部3は、フォーク要求7aを受信すると、フォーク要求7aに付随する子スレッド生成済信号及び隣接プロセッサの状態に基づいて、隣接する他プロセッサ1-jに対するフォークが可能か否かを判定し、可能ならば当該プロセッサ1-jに対してフォーク先アドレスを伴うスレッド開始要求7cを送信する一方、フォーク要求元のプロセッサ1-iに対しては、フォーク応答7bを返却する。この時点で初めてフォークが行われたことになり、フォーク応答7bを受信したプロセッサ1-iは、フォーク先のプロセッサ1-jのレジスタファイルに対して、親スレッドのレジスタファイルの全内容を通信バス6を通じてコピーするか、当該子スレッドで必要なレジスタの値だけをコピーすることにより、レジスタ継承を行う。

【0030】他方、プロセッサ1-iからのフォーク要求時、隣接するプロセッサ1-jに対するフォークが不可能ならば、スレッド管理部3は、今回のフォーク要求7aを廃棄する。これにより前記フォーク命令は無効化される。

【0031】図3を参照すると、スレッド管理部3の一例は、スレッド管理シーケンサ11とプロセッサ状態テーブル12とから構成される。プロセッサ状態テーブル12は、プロセッサ1-iと1対1に対応する状態エントリ13-i及び最古親ビット14-iを有する。個々の状態エントリ13-iは、対応するプロセッサ1-iがビジー状態か、フリー状態かを記録するために使用される。個々の最古親ビット14-iは、対応するプロセッサ1-iで実行中のスレッドが全プロセッサで実行中のスレッドの最も祖先のスレッドである場合、1にセットされ、それ以外は0にセットされる。並列プログラムの実行開始時点で、最初のスレッドを実行するプロセッサに対応する最古親ビットのみが1に初期設定され、以後、プログラムのスレッドの終了、生成に応じて最古親ビット14-iが適宜更新されていく。スレッド管理シーケンサ11は、このプロセッサ状態テーブル12を用いて各プロセッサ1-iにおけるスレッド生成、スレッド終了を管理する。プロセッサ1-iからフォーク要求7a、スレッド終了通知7dを受信した際のスレッド管理シーケンサ11の処理例を図4及び図5に示す。

【0032】図4を参照すると、スレッド管理シーケンサ11は、或るクロックのタイミングで何れかのプロセ

10

20

30

40

50

ッサ1-iからフォーク要求7aを受信すると、先ず、そのフォーク要求7aに付随する子スレッド生成済信号を調べる(ステップS1)。子スレッド生成済信号が子スレッドの生成済みを示す値1でないときは(ステップS1でNO)、隣接するプロセッサ1-jの状態をプロセッサ状態テーブル12で調べ、フリー状態であれば(ステップS2でYES)、フォーク可能なため、プロセッサ状態テーブル12における当該プロセッサ1-jに対応するエントリ13-jをフリー状態からビジー状態に更新し(ステップS3)、フォーク要求7aに付随するフォーク先アドレスを添えたスレッド開始要求7cをフォーク先プロセッサ1-jに送信すると共に、要求元のプロセッサ1-iに対してフォーク応答7bを返却する(ステップS4)。隣接するプロセッサ1-jがビジー状態であれば(ステップS2でNO)、プロセッサ1-jはプロセッサ1-iの現実行中スレッドからフォークされた子スレッド以外のスレッドを実行中であり、フォーク不可能なので、スレッド管理シーケンサ11は、当該フォーク要求7aを廃棄する(ステップS5)。

【0033】他方、フォーク要求7aに付随する子スレッド生成済信号が子スレッドの生成済みを示す値1のときは(ステップS1でYES)、隣接プロセッサ1-jは当該プロセッサ1-iで実行中のスレッドからフォークされた子スレッドを実行していることになり、それをキャンセルすることによりフォーク可能である。このため、ステップS4へ進み、フォーク要求7aに付随するフォーク先アドレスを添えたスレッド開始要求7cをフォーク先プロセッサ1-jに送信すると共に、要求元のプロセッサ1-iに対してフォーク応答7bを返却する。フォーク先プロセッサ1-jで実行中の子スレッドはスレッド開始要求7cによって後述するようにキャンセルされる。

【0034】図5を参照すると、スレッド管理シーケンサ11は、何れかのプロセッサ1-iからスレッド終了通知7dを受信すると、プロセッサ管理テーブル12における当該プロセッサ1-iに対応する最古親ビット14-iを参照し、当該プロセッサ1-iで実行中のスレッドが最古親スレッドであるか否かを判定する(ステップS11)。最古親スレッドであれば、対応する状態エントリ13-iをビジー状態からフリー状態に更新すると共に、対応する最古親ビット14-iを1から0に書き換え且つ隣接するプロセッサ14-jの最古親ビット14-jを0から1に書き換えることで、最古親スレッドを更新する(ステップS12)。そして、スレッド終了通知7dを出したプロセッサ1-iに対してスレッド終了許可7eを送信する(ステップS13)。他方、スレッド終了通知7dを出したプロセッサ1-iで実行中のスレッドが最古親スレッドでなければ(ステップS11でNO)、プロセッサ1-iで実行中のスレッドが最古

親スレッドになるまでスレッド終了許可を保留する(ステップS14)。

【0035】図6を参照すると、各々のプロセッサ1-iは、スレッド管理部3から送信されたスレッド開始要求7cに付随する開始アドレス値がセットされ、その後適宜歩進されるPC21と、PC21に従ってメモリ5からスレッドの命令をフェッチする命令フェッチユニット22と、フェッチされた命令をデコードし、実行する実行ユニット23と、汎用レジスタ24-0~24-mの集合であるレジスタファイル25と、フォーク先プロセッサに対して通信バス6経由でレジスタファイル25の内容を転送するレジスタ転送ユニット26と、フォーク命令実行時に実行ユニット23からスレッド管理部3に送信されるフォーク要求7aに付随するフォーク先アドレスを保存するレジスタ27と、フォーク要求7aに対するフォーク応答7bによってセットされるフォークドビット28と、PC21の値がレジスタ27に保存されたフォーク先アドレスと一致するか否かを判定する一致回路29と、フォークドビット28及び一致回路29の出力の論理積信号を実行ユニット23に出力するアンドゲート30と、仮実行用バッファ31とを含んで構成されている。フォークドビット28の出力は、フォーク要求7aに付随する子スレッド生成済信号としても利用される。

【0036】各々のプロセッサ1-iは、スレッド開始要求7cによって、それに付随する開始アドレスからスレッドの実行を開始する。スレッドを実行中であればそれをキャンセルして新たなスレッドの実行を開始する。実行ユニット23は、スレッドの仮実行状態が解かれるまでは、実行の取り消しがハードウェア上可能である範囲内で仮実行を行う。つまり共有メモリ5及び図示しないキャッシュメモリへの書き込みは抑制し、仮実行用バッファ31に対して書き込みを行う。スレッドの仮実行状態は、本実施例では、スレッド終了通知7dに対する応答としてスレッド終了許可7eを受信したときに解かれる。スレッドの仮実行状態が解かれると、実行ユニット23は仮実行用バッファ31の内容を共有メモリ5及び図示しないキャッシュメモリに反映する。仮実行用バッファ31を用いない通常の状態となる。

【0037】従来のマルチスレッド実行方法では、子スレッドの実行をキャンセルするには、親スレッドでスレッド破棄命令(abort)を実行する必要があったが、本実施例のプロセッサ1-iではスレッド開始要求7cが仮実行用バッファ31にキャンセル信号として与えられているため、新しい子スレッドを起動すれば既に実行中の子スレッドが自動的にキャンセルされるようになっており、親スレッドでスレッド破棄命令を実行する必要はない。また、本実施例では、スレッド終了許可7eを受信したときに仮実行状態を解除するようにしたため、従来のマルチスレッド実行方法におけるような投機

成功を子スレッドに通知する命令は必要ではない。

【0038】また、実行ユニット23は、フォークドビット28がセットされている状態においてPC21の値がレジスタ27に保存されたフォーク先アドレスと一致することによりアンドゲート30の出力が論理“1”になると、スレッドの処理を終了すべく、スレッド管理部3に対してスレッド終了通知7dを送信する。PC21の値がレジスタ27に保存されたフォーク先アドレスと一致しても、フォークドビット28がセットされていなければ、アンドゲート30の出力は論理“1”にならないため、実行ユニット23はPC21に従って命令の実行を継続する。

【0039】レジスタ転送ユニット26は、フォークドビット28がセットされるタイミングでフォーク先プロセッサへのレジスタ転送を開始する。レジスタ転送ユニット26は、例えば、通信バス6のバス幅によって一度に転送できる数のレジスタ毎に、レジスタファイル25のレジスタの値とレジスタ番号（レジスタアドレス）とをフォーク先プロセッサのレジスタファイルへ送信し、受信側のレジスタファイル25では該当するレジスタを書き換える。

【0040】スレッドの開始から終了までのプロセッサ1-iの処理の概要を図7に示す。スレッド管理部3からのスレッド開始要求7cに基づき、プロセッサ1-iで1つのスレッドの実行が開始される際、当該プロセッサ1-iのフォークドビット28がリセットされ、またスレッドを実行中であればそれがキャンセルされる（ステップS21）。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される（ステップS22）。

【0041】実行ユニット23でデコードされた命令がフォーク命令の場合（ステップS24でYES）、実行ユニット23はフォーク先アドレスをレジスタ27に保存（上書き）し（ステップS25）、このレジスタ27に保存したフォーク先アドレスとフォークドビット28の値とを伴ったフォーク要求7aをスレッド管理部3に送信する（ステップS26）。スレッド管理部3は、フォークドビット28の値が1か、0でも隣接プロセッサ1-jがフリー状態のときフォーク可能と判定し、要求元のプロセッサ1-iに対してフォーク応答7bを返却し、隣接プロセッサ1-jに対してはスレッド開始要求7cを送出する。フォーク応答7bを受信したプロセッサ1-iは、フォークドビット28を1にセットし、レジスタ転送ユニット26によって親スレッドのレジスタファイル25の内容を通信バス6経由でフォーク先プロセッサ1-jのレジスタファイルに転送するレジスタ継承操作を行う（ステップS30）。また、フォーク先プロセッサ1-jでは図7のステップS21以降の処理を実行する。

【0042】他方、スレッド管理部3は、フォークドビ

ット28の値が0で且つ隣接プロセッサ1-jがビジー状態のときフォーク不可能と判定し、プロセッサ1-iから送信されたフォーク要求7aを廃棄する。従って、プロセッサ1-iで実行された今回のフォーク命令は無効化され、当該フォーク命令による子スレッドのフォークは断念される。

【0043】プロセッサ1-iで命令の実行が進み、PC21の値がレジスタ27に保存されたフォーク先アドレスに一致すると（ステップS23でYES）、フォークドビット28がセットされていれば（ステップS27でYES）、アンドゲート30の出力が論理“1”となり、実行ユニット23に割り込みがかけられ、当該プロセッサ1-iはスレッド終了通知7dをスレッド管理部3に送信し（ステップS28）、スレッド管理部3からスレッド終了許可7eを受信した時点でスレッドの処理を終了する（ステップS29）。しかし、フォークドビット28がセットされていなければ、PC21に従って命令の実行を継続して実行する（ステップS22）。

【0044】本実施例のマルチスレッド実行方法の実行シーケンスの一例を図8（a）に示す。この実行シーケンスは同図（b）に示すような制御依存投機処理において投機が失敗した場合のシーケンスを示しており、プロセッサ#0からプロセッサ#1にフォークされた制御依存投機にかかる子スレッドは次の子スレッドのフォーク時にキャンセルされている。一方、図8（c）、（d）は同じような制御依存投機を従来のマルチスレッド実行方法で実行する際の実行シーケンスと制御依存投機処理を示しており、プロセッサ#0からプロセッサ#1にフォークした制御依存投機にかかる子スレッドをスレッド破棄命令（abort）によってキャンセルしてから、次の子スレッドをフォークしている。本実施例では、2度目のフォークで既に存在する子スレッドをキャンセルするため、並列化プログラムへのスレッド破棄命令の挿入が不要である。また、子スレッドの開始点の直前へのターム命令（term）の挿入が不要である。

【0045】本実施例のマルチスレッド実行方法の実行シーケンスの別の例を図9（a）に示す。この図9

（a）は、プロセッサ#0で実行しているスレッドの最初のフォーク命令Aの時点で、フォーク先プロセッサ#1がビジー状態の場合を想定している。本実施例では、このような場合は当該フォーク命令Aによるフォークが即断念される。このようにフォーク命令Aのフォークを断念してもプログラムの処理は正しく遂行される。これを図1（e）に示した並列化プログラムを例に説明すると、以下ようになる。図1（e）中のフォーク命令fork_th1がフォーク命令Aに、フォーク命令fork_th2がフォーク命令Bにそれぞれ対応するため、フォーク命令Aが無効化され、フォーク命令Bがフォークされた場合の実行シーケンスは図9（b）のようになる。プロセッサ#0では命令0、命令1がその順

に、プロセッサ#1では命令2、命令3、命令4がその順にそれぞれ実行されており、プログラムの処理は支障なく行える。

【0046】同様に図9(a)においてフォーク命令Bの時点で、なおもフォーク先プロセッサ#1がビジー状態であると、フォーク命令Bも無効化される。図1

(e)の並列化プログラムの場合、その実行シーケンスは図9(c)に示すようになり、プロセッサ#0において、命令0、命令1、命令2、命令3、命令4がこの順に逐次に実行されることになる。

【0047】次に、本実施例のマルチスレッド実行方法で実行される並列化プログラムの生成方法について説明する。

【0048】図10を参照すると、コンパイラ41は、逐次処理プログラム42を入力し、制御及びデータフロー解析部44によって逐次処理プログラム42の制御フロー及びデータフローを解析して、基本ブロック或いは複数の基本ブロックを並列化の単位、すなわちスレッドに分割し、次いで並列化コード挿入部45によって並列化のためのコードを挿入して、複数のスレッドに分割された並列化プログラム43を生成して出力する。

【0049】一般に並列化コードとしては、フォーク命令、ターム命令などがある。本実施例では並列プログラムの生成時に、フォーク点にフォーク命令が挿入される。しかし、子スレッドの開始点の直前に従来挿入されていたターム命令は挿入されない。また、図9(a)のフォーク命令Aのように本実施例ではフォーク不可能ならばフォーク命令が無効化されるため、コンパイラ41はフォーク命令が無効化されても逐次処理プログラム42の正常動作が保証される並列化プログラム43を生成する。一般に、並列化プログラム43中からフォーク命令(それに対応するターム命令があればそのターム命令も)を全て取り除いた状態の制御フローが逐次処理プログラム42の制御フローと等価であれば、逐次処理プログラム42の動作を保証できる並列化プログラム43となる。

【0050】以上説明したように本実施例によれば、フォーク1回制限の保証のない並列化プログラムであっても実行時にフォーク1回制限を保証することができる。また、後述する第4の実施例のように親スレッドの最初のフォーク命令時点で隣接プロセッサがビジー状態の場合にフォーク命令をウェイトさせると処理が中断するが、本実施例ではそのような場合でも処理の中断無しにプログラムの処理を進めることができる。更に、後述する第5の実施例のようにフォーク不可能な場合にレジスタファイルの内容を退避バッファに退避して後刻におけるフォークを可能にする構成では、ハードウェア量が退避バッファの分だけ増加し、また退避バッファもオペレーティングシステムのプロセス切り替え時の退避、復元対象となるためにプロセス切り替えオーバーヘッドが増

大するが、本実施例ではそのような問題も解消される。また後述する第6の実施例のように並列化プログラム中にターム命令を記述する必要がなく、プログラムサイズのコンパクト化による命令メモリの容量削減等が可能となる。

【0051】

【第1の発明の第2の実施例】第1の実施例では、親スレッドのフォーク点でフォーク可能でなければフォークを即断念したが、本実施例では、親スレッドのレジスタファイルが更新される前にフォーク先プロセッサがフリー状態になるとフォークを行う。以下、第1の実施例との相違点を中心に本実施例を説明する。

【0052】図11を参照すると、本実施例における並列プロセッサシステムの各々のプロセッサ1-iは、図6に示した構成に加えて、フォーク有効ビット33を含んで構成されている。フォーク有効ビット33は、実行ユニット23がフォーク命令を実行したときに出力するフォーク信号34でセットされ、スレッド管理部3から受信されるフォーク応答7b及び実行ユニット23が親スレッドのレジスタファイル25中の何れかのレジスタを更新したときに出力するレジスタ更新信号35によってリセットされる。フォーク有効ビット33の出力がスレッド管理部3に対するフォーク要求7aとなり、フォーク有効ビット33がセットされている間、フォーク要求7aが送出し続けられる。

【0053】前述の図4を参照すると、スレッド管理部3のスレッド管理シーケンス11は、或るクロックのタイミングでプロセッサ1-iからフォーク要求7aを受信した際、子スレッド生成済信号が0且つ隣接プロセッサ1-iがビジー状態のとき(ステップS2でNO)、当該フォーク要求7aは破棄したが、本実施例では、プロセッサ1-iはフォーク有効ビット33がセットされている間、フォーク要求7aを送出し続けているので、次のクロックのタイミングでスレッド管理部3がプロセッサ1-iからフォーク要求7aを再び受信することになり、図4の処理が繰り返される。即ち、フォーク点でフォーク不可能な場合、フォーク命令は保留にされ、フォーク可能となった時点で実行されることになる。

【0054】スレッドの開始から終了までのプロセッサ1-iの処理の概要を図12に示す。スレッド管理部3からのスレッド開始要求7cに基づき、プロセッサ1-iで1つのスレッドの実行が開始される際、当該プロセッサ1-iのフォークドビット28及びフォーク有効ビット33がリセットされ、またスレッドを実行中であればそれがキャンセルされる(ステップS21)。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される(ステップS22)。

【0055】実行ユニット23でデコードされた命令がフォーク命令の場合(ステップS24でYES)、実行ユニット23はフォーク先アドレスをレジスタ27に保

存し（ステップS25）、フォーク信号34によってフォーク有効ビット33をセットすることにより（ステップS26）、フォーク先アドレスとフォークドビット28の値とを伴ったフォーク要求7aをスレッド管理部3に送信する。また、実行ユニット23はレジスタファイル25中の何れかのレジスタを更新すると（ステップS31でYES）、レジスタ更新信号35を出力してフォーク有効ビット33をリセットする（ステップS32）。従って、プロセッサ1-iからは、フォーク命令実行時点からレジスタファイル25が最初に更新される迄の期間中、フォーク要求7aがスレッド管理部3に送出し続けられる。

【0056】スレッド管理部3は、プロセッサ1-iからフォーク要求7aが送出されている期間内で、隣接プロセッサ1-jに対するフォークが可能になると、プロセッサ1-iにフォーク応答7bを送信すると同時に他プロセッサ1-jに対してスレッド開始要求7cを送出することで子スレッドをフォークする（ステップS4）。プロセッサ1-iに出されたフォーク応答7bによって、フォークドビット28がセットされると共にフォーク有効ビット33はリセットされ、またレジスタ転送ユニット26によってレジスタファイル25の内容がフォーク先プロセッサ1-jに転送される（ステップS30）。なお、レジスタ転送ユニット26によるレジスタファイル25の転送中、実行ユニット23からレジスタファイル25への書き込みは待たされる。

【0057】プロセッサ1-iで命令の実行が進み、PC21の値がレジスタ27に保存されたフォーク先アドレスに一致したときの動作は第1の実施例と同様である。

【0058】本実施例のマルチスレッド実行方法の実行シーケンスの一例を図13に示す。フォーク命令Aから下に延びる矢印は、プロセッサ#0においてレジスタファイル25が全く更新されていない期間を示す。この図13は、プロセッサ#0で実行しているスレッドの最初のフォーク命令Aの時点で、フォーク先プロセッサ#1がビジー状態であったが、プロセッサ#0のレジスタファイル25が全く更新されていない期間内にフォーク先プロセッサ#1がフリー状態になった場合を想定している。本実施例では、このような場合はフォーク命令Aによるフォークが行われる。

【0059】以上説明したように本実施例によれば、第1の実施例と同様な効果が得られると共に、親スレッドの実行開始後、フォーク命令の時点でフォークできなくても、レジスタファイルが更新される前にフォーク先プロセッサがフリー状態になればフォークを行うため、第1の実施例に比べてフォークされる可能性が高まり、スレッド実行の並列度が向上する。

【0060】

【第1の発明の第3の実施例】第2の実施例では、親ス

レッドのフォーク点でフォーク可能でなければフォークを一旦保留にし、親スレッドのレジスタファイルが更新される前に子スレッドの実行を開始できる他プロセッサが生じなかった場合に当該フォークを断念したが、本実施例では、親スレッドのレジスタファイルが更新されても、その更新が子スレッドに継承すべきレジスタでなければフォークを行う。以下、第2の実施例との相違点を中心に本実施例を説明する。

【0061】図14を参照すると、本実施例における並列プロセッサシステムの各々のプロセッサ1-iは、図11に示した構成に加えて、レジスタファイル25の各レジスタ24-k（k=0~m）に1対1に対応し、対応するレジスタ24-kが子スレッドへ継承すべきレジスタであるときに限りセットされるクリエイティブビット36-kと、各レジスタ24-kに1対1に対応し、対応するレジスタ24-kのクリエイティブビット36-kの出力と実行ユニット23がレジスタ24-kを更新したときに出力するレジスタ更新信号37-kとを入力とするアンドゲート38-kと、アンドゲート38-kの出力の論理和信号であるフォーク無効信号40を出力するオアゲート39とを含んで構成されている。そして、図14のレジスタ更新信号35に代えて、フォーク無効信号40がフォーク有効ビット33にリセット信号として出力されている。また、各クリエイティブビット36-kの値がレジスタ転送ユニット26に出力されており、レジスタ転送ユニット26はレジスタファイル25のレジスタ24-kのうち、対応するクリエイティブビット36-kがセットされているレジスタのみをフォーク先プロセッサのレジスタファイルに転送するように構成されている。

【0062】スレッドの開始から終了までのプロセッサ1-iの処理の概要を図15に示す。スレッド管理部3からのスレッド開始要求7cに基づき、プロセッサ1-iで1つのスレッドの実行が開始される際、当該プロセッサ1-iのフォークドビット28及びフォーク有効ビット33がリセットされ、またスレッドを実行中であればそれがキャンセルされる（ステップS21）。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される（ステップS22）。

【0063】実行ユニット23でデコードされた命令がフォーク命令の場合（ステップS24でYES）、実行ユニット23はフォーク先アドレスをレジスタ27に保存すると共に、全てのクリエイティブビット36-kのセットアップを行う（ステップS25）。つまり、レジスタファイル25のレジスタ24-kのうち、子スレッドに継承すべきレジスタに対応するクリエイティブビット36-kはセットし、継承する必要のないレジスタに対応するクリエイティブビット36-kはリセットされたままにする。そして、フォーク信号34によってフォーク有効ビット33をセットすることにより（ステップS26）、フォーク先アドレスとフォークドビット28の値とを伴

ったフォーク要求 7 a をスレッド管理部 3 に送信する。また、実行ユニット 2 3 はレジスタファイル 2 5 中のレジスタ 2 4 - k を更新すると (ステップ S 3 1 で YES)、その更新したレジスタ 2 4 - k に対応するレジスタ更新信号 3 7 - k を論理 “1” とする。これにより、若し更新されたレジスタ 2 4 - k が子スレッドへ継承すべきレジスタであった場合、そのレジスタ 2 4 - k に対応するアンドゲート 3 8 - k の出力が論理 “1” となり、オアゲート 3 9 からフォーク無効信号 4 0 が出力されてフォーク有効ビット 3 3 がリセットされる (ステップ S 3 2)。つまり、プロセッサ 1 - i からは、フォーク命令実行時点からレジスタファイル 2 5 中の子スレッドへの継承レジスタの何れかが最初に更新される迄の期間中、フォーク要求 7 a がスレッド管理部 3 に送出し続けられる。

【0064】スレッド管理部 3 は、プロセッサ 1 - i からフォーク要求 7 a が送出されている期間内で、隣接プロセッサ 1 - j に対するフォークが可能になると、プロセッサ 1 - i にフォーク応答 7 b を送信すると同時に他プロセッサ 1 - j に対してスレッド開始要求 7 c を送出することで子スレッドをフォークする (ステップ S 4)。プロセッサ 1 - i に出されたフォーク応答 7 b によって、フォークドビット 2 8 がセットされると共にフォーク有効ビット 3 3 はリセットされ、またレジスタ転送ユニット 2 6 によってレジスタファイル 2 5 のレジスタの内、少なくとも子スレッドに継承すべきレジスタがフォーク先プロセッサ 1 - j に転送される (ステップ S 3 0)。

【0065】プロセッサ 1 - i で命令の実行が進み、PC 2 1 の値がレジスタ 2 7 に保存されたフォーク先アドレスに一致したときの動作は第 2 の実施例と同様である。

【0066】本実施例のマルチスレッド実行方法では、図 1 3 中のフォーク命令 A から下に述べる矢印は、プロセッサ # 0 においてレジスタファイル 2 5 のレジスタの内、子スレッドに継承すべきレジスタが全く更新されていない期間となる。従って、第 1 の実施例は勿論のこと、第 2 の実施例に比べてもフォークの可能性をより高めることができる。

【0067】本実施例では、親スレッドのフォーク点で子スレッドへ継承すべきレジスタが判明している必要がある。このため、図 1 0 に示したコンパイラ 4 1 における制御及びデータフロー解析部 4 4 では、フォークする子スレッド毎に、親スレッドから子スレッドへ継承すべきレジスタを調査し、並列化コード挿入部 4 5 ではその調査結果に基づいて、子スレッドへ継承すべきレジスタを指定する記述を並列化プログラム 4 3 に挿入する。子スレッドへ継承すべきレジスタの指定は、フォーク命令で指定する方法、フォーク命令とは別の専用の命令で指定する方法などが利用できる。

【0068】なお、本実施例におけるレジスタ転送ユニット 2 6 は、クリエイトビット 3 6 - k を参照することにより、親スレッドのレジスタファイル 2 5 のうち子スレッドに継承すべきレジスタだけをフォーク先プロセッサのレジスタファイルに転送するようにしたが、別の実施例として、レジスタファイル 2 5 の先頭のレジスタから順に所定の順番でレジスタの転送を行うシーケンスを開始し、クリエイトビット 3 6 - k がセットされているレジスタの全ての転送が完了した時点で転送シーケンスを停止するようにしても良い。この方法では、子スレッドに継承する必要のないレジスタも転送される場合があるが、転送シーケンスが簡素化される利点がある。勿論、別の実施例として、クリエイトビット 3 6 - k を一切参照せずに常に全レジスタを転送するようにレジスタ転送ユニット 2 6 が構成されていても良い。更に、子スレッドに継承すべきレジスタでも、フォーク先プロセッサの当該レジスタの値がフォーク時点で既に親スレッド側と同じ値になっている場合にはあえて転送する必要がない点に着目して、子スレッドに継承すべきレジスタのうち、親スレッド側と異なる値になっているレジスタを検出し、この検出したレジスタだけをレジスタ転送ユニット 2 6 からフォーク先プロセッサに転送するようにしても良い。

【0069】以上説明したように本実施例によれば、第 2 の実施例と同様な効果が得られると共に、親スレッドのレジスタファイルの更新があっても、その更新が子スレッドに継承すべきレジスタでなければフォークを行うため、第 2 の実施例に比べてフォークされる可能性をより高めることができ、従ってスレッド実行の並列度をより向上することができる。

【0070】

【第 1 の発明の第 4 の実施例】本実施例は、親スレッドの実行開始後、フォーク命令の時点で隣接プロセッサがビジー状態であった場合、隣接プロセッサがフリー状態になるまでフォーク命令の実行をウエイトするようにした点で、第 1 乃至第 3 の実施例と相違する。以下、第 2 の実施例との相違点を中心に本実施例を説明する。

【0071】図 1 6 を参照すると、本実施例における並列プロセッサシステムの各々のプロセッサ 1 - i は、図 1 1 に示した構成とほぼ同様な構成を有するが、スレッド管理部 3 から送信されるフォーク応答 7 b が実行ユニット 2 3 にも入力されており、実行ユニット 2 3 はフォーク命令の実行時、フォーク応答 7 b が返却されるまでフォーク命令の実行をウエイトする点で相違する。また、フォーク有効ビット 3 3 を実行ユニット 2 3 からリセットするレジスタ更新信号 3 5 は存在しない。

【0072】スレッドの開始から終了までのプロセッサ 1 - i の処理の概要を図 1 7 に示す。スレッド管理部 3 からのスレッド開始要求 7 c に基づき、プロセッサ 1 - i で 1 つのスレッドの実行が開始される際、当該プロセ

ッサ1-iのフォークドビット28及びフォーク有効ビット33がリセットされ、またスレッドを実行中であればそれがキャンセルされる(ステップS21)。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される(ステップS22)。

【0073】実行ユニット23でデコードされた命令がフォーク命令の場合(ステップS24でYES)、実行ユニット23はフォーク先アドレスをレジスタ27に保存し(ステップS25)、フォーク信号34によってフォーク有効ビット33をセットすることにより(ステップS26)、フォーク先アドレスとフォークドビット28の値とを伴ったフォーク要求7aをスレッド管理部3に送信する。そして、スレッド管理部3からフォーク応答7bが返却されるのを待つ(ステップS41)。

【0074】スレッド管理部3は、プロセッサ1-iからフォーク要求7aが送出されている期間内で、隣接プロセッサ1-jに対するフォークが可能になると、プロセッサ1-iにフォーク応答7bを送信すると同時に他プロセッサ1-jに対してスレッド開始要求7cを送出することで子スレッドをフォークする(ステップS44)。プロセッサ1-iは、スレッド管理部3からフォーク応答7bを受信すると、フォークドビット28をセットすると共にフォーク有効ビット33をリセットし、レジスタ転送ユニット26によってレジスタファイル25の内容をフォーク先プロセッサ1-jに転送する(ステップS42)。

【0075】プロセッサ1-iで命令の実行が進み、PC21の値がレジスタ27に保存されたフォーク先アドレスに一致したときの動作は第2の実施例と同様である。

【0076】

【第1の発明の第5の実施例】本実施例は、親スレッドの実行開始後、フォーク命令の時点で隣接プロセッサがビジー状態の場合、レジスタファイルの内容を退避させ、隣接プロセッサがフリー状態になった時点で前記退避した情報に基づいて子スレッドのフォークを行うようにした点で、第1乃至第4の実施例と相違する。以下、第4の実施例との相違点を中心に本実施例を説明する。

【0077】図18を参照すると、本実施例における並列プロセッサシステムの各々のプロセッサ1-iは、図16に示した構成に加えて、退避バッファ41と、退避バッファ有効ビット42と、実行ユニット23から出力される退避信号43によって起動されるとレジスタファイル25の内容を退避バッファ41に退避する退避ユニット44とを備え、レジスタ転送ユニット26はレジスタファイル25及び退避バッファ42に接続されている。退避バッファ有効ビット42は、実行ユニット23から出力されるフォーク信号34によってリセットされ、退避信号43によってセットされ、また退避バッファ41に基づくフォークが行われた場合にリセットされ

る。

【0078】スレッドの開始から終了までのプロセッサ1-iの処理の概要を図19に示す。スレッド管理部3からのスレッド開始要求7cに基づき、プロセッサ1-iで1つのスレッドの実行が開始される際、当該プロセッサ1-iのフォークドビット28及びフォーク有効ビット33がリセットされ、またスレッドを実行中であればそれがキャンセルされる(ステップS21)。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される(ステップS22)。

【0079】実行ユニット23でデコードされた命令がフォーク命令の場合(ステップS24でYES)、実行ユニット23はフォーク先アドレスをレジスタ27に保存し(ステップS25)、フォーク信号34によってフォーク有効ビット33をセットすることによりフォーク先アドレス及びフォークドビット28の値を伴ったフォーク要求7aをスレッド管理部3に送信する(ステップS26)。このとき退避バッファ有効ビット42がリセットされる。そして、スレッド管理部3から所定の時間内にフォーク応答7bが返却された場合(ステップS43でYES)、フォークドビット28をセットすると共にフォーク有効ビット33をリセットし、レジスタ転送ユニット26によってレジスタファイル25の内容をフォーク先プロセッサ1-jに転送する(ステップS44)。

【0080】他方、スレッド管理部3から所定の時間内にフォーク応答7bが返却されなかった場合(ステップS43でNO)、退避信号43によって退避ユニット44を起動することによりレジスタファイル25の内容を退避バッファ41へ退避させ、退避バッファ有効ビット42をセットする(ステップS45)。この退避バッファ有効ビット42がセットされている間に、スレッド管理部3からフォーク応答7bを受信すると(ステップS46、S47でYES)、レジスタ転送ユニット26により退避バッファ41に退避されているレジスタファイル25の内容をフォーク先プロセッサ1-jに転送する(ステップS48)。このとき、フォークドビット28がセットされ、フォーク有効ビット33及び退避バッファ有効ビット42がリセットされる。なお、次のフォーク命令が実行される迄に前回のフォーク命令に対するフォーク応答7bがスレッド管理部3から返却されなかった場合、今回のフォーク命令の実行により退避バッファ有効ビットはリセットされるので(ステップS26)、前回のフォーク命令は結果的に無効化される。

【0081】プロセッサ1-iで命令の実行が進み、PC21の値がレジスタ27に保存されたフォーク先アドレスに一致したときの動作は第4の実施例と同様である。

【0082】

【第1の発明の第6の実施例】第1乃至第5の実施例で

は、各プロセッサ 1-i はプログラムカウンタの値が有効な子スレッドの開始アドレスと一致したときにスレッドの処理を終了したが、本実施例では、各プロセッサは、各フォーク命令に対応して並列化プログラム中のフォーク先アドレスの直前に挿入されているターム命令のうち、有効な子スレッドをフォークしたフォーク命令に対応するターム命令によってスレッドの処理を終了する。以下、第 1 の実施例との相違点を中心に本実施例を説明する。

【0083】図 20 を参照すると、本実施例における並列プロセッサシステムの各々のプロセッサ 1-i は、図 6 に示した構成のプロセッサにおける一致回路 29 及びアンドゲート 30 が省略され、フォークドビット 28 の出力が実行ユニット 23 に入力され、またスレッド ID を保持するレジスタ 45 が設けられている。このレジスタ 45 は、スレッド管理部 3 からのスレッド開始要求 7c に付随するスレッド ID が初期設定され、プロセッサ 1-i でフォーク命令が実行される毎に、そのフォーク命令で指定されたスレッド ID がセットされる。更に、スレッド管理部 3 へのフォーク要求 7a には、レジスタ 27 に保存されたフォーク先アドレスとレジスタ 45 に保存されたスレッド ID とフォークドビット 28 の値とが付随する。

【0084】スレッド管理部 3 は、図 4 のステップ S4 でフォーク先プロセッサ 1-j へスレッド開始要求 7c を送信する際、プロセッサ 1-i からのフォーク要求 7a に付随するフォーク先アドレス及びスレッド ID を一緒に送信する。

【0085】スレッドの開始から終了までのプロセッサ 1-i の処理の概要を図 21 に示す。スレッド管理部 3 からのスレッド開始要求 7c に基づき、プロセッサ 1-i で 1 つのスレッドの実行が開始される際、当該プロセッサ 1-i のフォークドビット 28 がリセットされ、スレッド開始要求 7c に付随するスレッド ID がレジスタ 45 にセットされ、またスレッドを実行中であればそれがキャンセルされる（ステップ S21）。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される（ステップ S22）。

【0086】実行ユニット 23 でデコードされた命令がフォーク命令の場合（ステップ S24 で YES）、実行ユニット 23 はフォーク先アドレスをレジスタ 27 に保存すると共にそのフォーク命令で指定されたスレッド ID をレジスタ 45 に保存（上書き）し（ステップ S25）、レジスタ 27 に保存したフォーク先アドレス、レジスタ 45 に保存したスレッド ID 及びフォークドビット 28 の値を伴ったフォーク要求 7a をスレッド管理部 3 に送信する（ステップ S26）。スレッド管理部 3 は、隣接プロセッサ 1-j に対してフォークが可能ならば、要求元のプロセッサ 1-i に対してフォーク応答 7b を返却し、隣接プロセッサ 1-j に対してはフォーク

先アドレス及びスレッド ID を付加したスレッド開始要求 7c を送出する。フォーク応答 7b を受信したプロセッサ 1-i は、フォークドビット 28 を 1 にセットし、レジスタ転送ユニット 26 によって親スレッドのレジスタファイル 25 の内容を通信バス 6 経由でフォーク先プロセッサ 1-j のレジスタファイルに転送するレジスタ継承操作を行う（ステップ S30）。また、フォーク先プロセッサ 1-j では図 21 のステップ S21 以降の処理を実行する。

【0087】他方、スレッド管理部 3 は、隣接プロセッサ 1-j に対してフォークが不可能ならば、プロセッサ 1-i から送信されたフォーク要求 7a を廃棄する。従って、プロセッサ 1-i で実行された今回のフォーク命令は無効化され、当該フォーク命令による子スレッドのフォークは断念される。

【0088】実行ユニット 23 でデコードされた命令がターム命令の場合（ステップ S51 で YES）、実行ユニット 23 は、フォークドビット 28 が 1 にセットされており且つ当該ターム命令で指定されたスレッド ID がレジスタ 45 に保存されているスレッド ID と一致した場合（ステップ S52、S53 で YES）、当該ターム命令を実行することにより、スレッド終了通知 7d を送出し（ステップ S28）、スレッド管理部 3 からスレッド終了許可 7e を受信した時点でスレッドの処理を終了する（ステップ S29）。しかし、フォークドビット 28 が 1 にセットされていないか、ターム命令で指定されたスレッド ID がレジスタ 45 に保存されているスレッド ID と一致しない場合は、当該ターム命令を無効にし、PC 21 に従って命令の実行を継続して実行する（ステップ S22）。

【0089】本実施例では、各プロセッサはターム命令によってスレッドの処理を終了するため、図 10 のコンパイラ 41 における並列化コード挿入部 45 は、子スレッドの開始点の直前に、当該子スレッドをフォークするフォーク命令に付加したスレッド ID と同じスレッド ID を持つターム命令を挿入する。

【0090】本実施例と同様に、第 2 乃至第 5 の実施例において、各プロセッサが各フォーク命令に対応して並列化プログラム中のフォーク先アドレスの直前に挿入されているターム命令のうち、有効な子スレッドをフォークしたフォーク命令に対応するターム命令によってスレッドの処理を終了するように変形することができる。

【0091】以上説明したように第 1 の発明によれば、フォーク 1 回制限の保証のない並列化プログラムであってもフォーク 1 回モデルによるマルチスレッド実行が可能になると共に以下のような効果が得られる。

(1) 親スレッドのフォーク命令毎に親スレッドから生成された子スレッドが既に存在する場合にはその子スレッドをキャンセルすることで、プログラム実行時にフォーク 1 回制限を保証する為、動作が決定的であり、制御

が簡単である。

(2) 後述する第2の発明に比べてスレッドの粒度が小さくなり、粒度を揃え易い。

(3) スレッド破棄命令 (abort) 無しで、制御依存投機処理が可能になる。

【0092】次に第2の発明の実施例について図面を参照して詳細に説明する。

【0093】

【第2の発明の第1の実施例】本実施例にかかる並列プロセッサシステムは、図2に示した並列プロセッサシステムと同様な構成を有する。但し、各プロセッサ1-iにおける仮実行用バッファを用いたスレッド実行の取り消し(キャンセル)機能は必須でない。また、各プロセッサ1-iは、信号線2-iを通じてスレッド管理部3に対してフォーク要求7aを送信する際、子スレッドのフォーク先アドレス(開始PC値)を付随させるが、子スレッド生成済み信号は付随させない。

【0094】図22を参照すると、本実施例の場合、スレッド管理部3のスレッド管理シーケンサ11は、或るクロックのタイミングで何れかのプロセッサ1-iからフォーク要求aを受信すると、隣接するプロセッサ1-jの状態をプロセッサ状態テーブル12で調べ、フリー状態であれば(ステップS61でYES)、フォーク可能なため、プロセッサ状態テーブル12における当該プロセッサ1-jに対応するエントリ13-jをフリー状態からビジー状態に更新し(ステップS62)、フォーク要求7aに付随するフォーク先アドレスを添えたスレッド開始要求7cをフォーク先プロセッサ1-jに送信すると共に、要求元のプロセッサ1-iに対してフォーク応答7bを返却する(ステップS63)。隣接するプロセッサ1-jがビジー状態であれば(ステップS61でNO)、フォーク不可能なので、スレッド管理シーケンサ11は、当該フォーク要求7aを廃棄する(ステップS64)。スレッド管理シーケンサ11が何れかのプロセッサ1-iからスレッド終了通知7dを受信したときの処理は図5と同じである。

【0095】図23を参照すると、各々のプロセッサ1-iは、スレッド管理部3から送信されたスレッド開始要求7cに付随する開始アドレス値がセットされ、その後適宜歩進されるPC21と、PC21に従ってメモリ5からスレッドの命令をフェッチする命令フェッチユニット22と、フェッチされた命令をデコードし、実行する実行ユニット23と、汎用レジスタ24-0~24-mの集合であるレジスタファイル25と、フォーク先プロセッサに対して通信バス6経由でレジスタファイル25の内容を転送するレジスタ転送ユニット26と、フォーク命令実行時に実行ユニット23からスレッド管理部3に送信されるフォーク要求7aに付随するフォーク先アドレスを保存するレジスタ27と、フォーク要求7aに対するフォーク応答7bによってセットされるフォ

ークドビット28と、PC21の値がレジスタ27に保存されたフォーク先アドレスと一致するか否かを判定する一致回路29と、フォークドビット28及び一致回路29の出力の論理積信号を実行ユニット23に出力するアンドゲート30とを含んで構成され、フォークドビット28の値は実行ユニット23にも入力されている。

【0096】各々のプロセッサ1-iは、スレッド開始要求7cによって、それに付随する開始アドレスからスレッドの実行を開始する。また、実行ユニット23は、フォーク命令のデコード時点で、フォークドビット28がセットされているか否かを調べ、セットされていなければ当該フォーク命令を実行するが、セットされていれば当該フォーク命令を無効にする。さらに実行ユニット23は、フォークドビット28がセットされている状態においてPC21の値がレジスタ27に保存されたフォーク先アドレスと一致することによりアンドゲート30の出力が論理“1”になると、スレッドの処理を終了すべく、スレッド管理部3に対してスレッド終了通知7dを送信する。PC21の値がレジスタ27に保存されたフォーク先アドレスと一致しても、フォークドビット28がセットされていなければ、アンドゲート30の出力は論理“1”にならないため、実行ユニット23はPC21に従って命令の実行を継続する。

【0097】レジスタ転送ユニット26は、フォークドビット28がセットされるタイミングでフォーク先プロセッサへのレジスタ転送を開始する。レジスタ転送ユニット26は、例えば、通信バス6のバス幅によって一度に転送できる数のレジスタ毎に、レジスタファイル25のレジスタの値とレジスタ番号(レジスタアドレス)とをフォーク先プロセッサのレジスタファイルへ送信し、受信側のレジスタファイル25では該当するレジスタを書き換える。

【0098】スレッドの開始から終了までのプロセッサ1-iの処理の概要を図24に示す。スレッド管理部3からのスレッド開始要求7cに基づき、プロセッサ1-iで1つのスレッドの実行が開始される際、当該プロセッサ1-iのフォークドビット28がリセットされる(ステップS71)。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される(ステップS72)。

【0099】実行ユニット23でデコードされた命令がフォーク命令の場合(ステップS74でYES)、実行ユニット23はフォークドビット28が1にセットされていれば(ステップS75でYES)、つまり既に1回フォークを行っていれば、今回のフォーク命令を無効化し、次の命令を実行する(ステップS72)。他方、フォークドビット28が0であれば(ステップS75でNO)、つまり親スレッドの実行後、フォークを1回も行っていないければ、フォーク先アドレスをレジスタ27に保存し(ステップS76)、このレジスタ27に保存し

たフォーク先アドレスを伴ったフォーク要求 7 a をスレッド管理部 3 に送信する (ステップ S 7 7)。

【0100】スレッド管理部 3 は、図 2 2 を参照して説明したように隣接プロセッサ 1-j に対してフォークが可能ならば、要求元のプロセッサ 1-i に対してフォーク応答 7 b を返却し、隣接プロセッサ 1-j に対してはスレッド開始要求 7 c を送出する。フォーク応答 7 b を受信したプロセッサ 1-i は、フォークドビット 2 8 を 1 にセットし、レジスタ転送ユニット 2 6 によって親スレッドのレジスタファイル 2 5 の内容を通信バス 6 経由でフォーク先プロセッサ 1-j のレジスタファイルに転送するレジスタ継承操作を行う (ステップ S 8 1)。また、フォーク先プロセッサ 1-j では図 2 4 のステップ S 7 1 以降の処理を実行する。

【0101】他方、スレッド管理部 3 は、隣接プロセッサ 1-j に対してフォークが不可能ならば、プロセッサ 1-i から送信されたフォーク要求 7 a を廃棄する (ステップ S 6 4)。従って、プロセッサ 1-i で実行された今回のフォーク命令は無効化され、当該フォーク命令による子スレッドのフォークは断念される。

【0102】プロセッサ 1-i で命令の実行が進み、PC 2 1 の値がレジスタ 2 7 に保存されたフォーク先アドレスに一致すると (ステップ S 7 3 で YES)、フォークドビット 2 8 がセットされていれば (ステップ S 7 8 で YES)、アンドゲート 3 0 の出力が論理 “1” となり、実行ユニット 2 3 に割り込みがかかり、当該プロセッサ 1-i はスレッド終了通知 7 d をスレッド管理部 3 に送信し (ステップ S 7 9)、スレッド管理部 3 からスレッド終了許可 7 e を受信した時点でスレッドの処理を終了する (ステップ S 8 0)。しかし、フォークドビット 2 8 がセットされていなければ、PC 2 1 に従って命令の実行を継続して実行する (ステップ S 7 2)。

【0103】本実施例のマルチスレッド実行方法の実行シーケンスの例を図 2 5 (a) に示す。この例は、プロセッサ # 0 で実行している親スレッドの最初のフォーク命令 A の時点で、フォーク先プロセッサ # 1 がフリー状態の場合を想定している。この場合、親スレッド中のフォーク命令 B などフォーク命令 A 以外の全てのフォーク命令は無効化される。

【0104】本実施例のマルチスレッド実行方法の実行シーケンスの別の例を図 2 5 (b) に示す。この例は、プロセッサ # 0 で実行しているスレッドの最初のフォーク命令 A の時点で、フォーク先プロセッサ # 1 がビジー状態の場合を想定しており、フォーク命令 A は無効化されている。また、次のフォーク命令 B の時点でフォーク先プロセッサ # 1 がフリー状態になっていた為、フォークが行われている例を示す。この場合、親スレッド中のフォーク命令 B 以降の全てのフォーク命令が無効化される。

【0105】図 2 5 (a) の実行シーケンスを図 1

(e) に示した並列化プログラムに当てはめると、図 1 (e) 中のフォーク命令 fork th 1 がフォーク命令 A に、フォーク命令 fork th 2 がフォーク命令 B にそれぞれ対応する。このためフォーク命令 A が実行され、フォーク命令 B が無効化された場合の実行シーケンスは図 2 5 (c) のようになる。プロセッサ # 0 では命令 0、命令 1、命令 2 がその順に、プロセッサ # 1 では命令 3、命令 4 がその順にそれぞれ実行されており、プログラムの処理は支障なく行える。

10 【0106】また図 2 5 (b) の実行シーケンスを図 1 (e) に示した並列化プログラムに当てはめると、図 2 5 (d) のようになる。プロセッサ # 0 では命令 0、命令 1 がその順に、プロセッサ # 1 では命令 2、命令 3、命令 4 がその順にそれぞれ実行されており、プログラムの処理は支障なく行える。

【0107】なお、フォーク命令 A 及び B が無効化されると実行シーケンスは図 2 5 (e) に示すようになり、プロセッサ # 0 において、命令 0、命令 1、命令 2、命令 3、命令 4 がこの順に逐次に行われることになる。

20 【0108】本実施例のマルチスレッド実行方法で実行される並列化プログラムの生成方法は第 1 の発明の第 1 の実施例と同じである。

【0109】以上説明したように本実施例によれば、フォーク 1 回制限の保証のない並列化プログラムであっても実行時にフォーク 1 回制限を保証することができる。また、後述する第 4 の実施例のように親スレッドの最初のフォーク命令時点で隣接プロセッサがビジー状態の場合にフォーク命令をウェイトさせると処理が中断するが、本実施例ではそのような場合でも処理の中断無しにプログラムの処理を進めることができる。更に、後述する第 5 の実施例のようにフォーク不可能な場合にレジスタファイルの内容を退避バッファに退避して後刻におけるフォークを可能にする構成では、ハードウェア量が退避バッファの分だけ増加し、また退避バッファもオペレーティングシステムのプロセス切り替え時の退避、復元対象となるためにプロセス切り替えオーバーヘッドが増大するが、本実施例ではそのような問題も解消される。また後述する第 6 の実施例のように並列化プログラム中にターム命令を記述する必要がなく、プログラムサイズ

30 のコンパクト化による命令メモリの容量削減等が可能となる。

【0110】

【第 2 の発明の第 2 の実施例】第 1 の実施例では、親スレッドのフォーク時点でフォーク可能でなければフォークを即断念したが、本実施例では、親スレッドのレジスタファイルが更新される前にフォーク先プロセッサがフリー状態になるとフォークを行う。以下、第 1 の実施例との相違点を中心に本実施例を説明する。

【0111】図 2 6 を参照すると、本実施の形態における並列プロセッサシステムの各々のプロセッサ 1-i

は、図 23 に示した構成に加えて、フォーク有効ビット 33 を含んで構成されている。フォーク有効ビット 33 は、実行ユニット 23 がフォーク命令を実行したときに出力するフォーク信号 34 でセットされ、スレッド管理部 3 から受信されるフォーク応答 7 b 及び実行ユニット 23 が親スレッドのレジスタファイル 25 中の何れかのレジスタを更新したときに出力するレジスタ更新信号 35 によってリセットされる。フォーク有効ビット 33 の出力がスレッド管理部 3 に対するフォーク要求 7 a となり、フォーク有効ビット 33 がセットされている間、フォーク要求 7 a が送出し続けられる。

【0112】前述の図 22 を参照すると、スレッド管理部 3 のスレッド管理シーケンサ 11 は、或るクロックのタイミングでプロセッサ 1-i からフォーク要求 7 a を受信した際、隣接プロセッサ 1-i がビジー状態のとき、当該フォーク要求 7 a は破棄したが（ステップ S 64）、本実施例では、プロセッサ 1-i はフォーク有効ビット 33 がセットされている間、フォーク要求 7 a を送出し続けているので、次回のクロックのタイミングでスレッド管理部 3 がプロセッサ 1-i からフォーク要求 7 a を再び受信することになり、図 22 の処理が繰り返される。即ち、フォーク時点でフォーク不可能な場合、フォーク命令は保留にされ、フォーク可能となった時点で実行されることになる。

【0113】スレッドの開始から終了までのプロセッサ 1-i の処理の概要を図 27 に示す。スレッド管理部 3 からのスレッド開始要求 7 c に基づき、プロセッサ 1-i で 1 つのスレッドの実行が開始される際、当該プロセッサ 1-i のフォークドビット 28 及びフォーク有効ビット 33 がリセットされる（ステップ S 71）。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される（ステップ S 72）。

【0114】実行ユニット 23 でデコードされた命令がフォーク命令の場合（ステップ S 74 で YES）、実行ユニット 23 はフォークドビット 28 が 1 にセットされていれば（ステップ S 75 で YES）、今回のフォーク命令を無効化し、次の命令を実行する（ステップ S 72）。他方、フォークドビット 28 が 0 であれば（ステップ S 75 で NO）、フォーク先アドレスをレジスタ 27 に保存し（ステップ S 76）、フォーク有効ビット 33 をセットすることにより、レジスタ 27 に保存したフォーク先アドレスを伴ったフォーク要求 7 a をスレッド管理部 3 に送信する（ステップ S 77）。また、実行ユニット 23 はレジスタファイル 25 中の何れかのレジスタを更新すると（ステップ S 82 で YES）、レジスタ更新信号 35 を出力してフォーク有効ビット 33 をリセットする（ステップ S 83）。従って、プロセッサ 1-i からは、フォーク命令実行時点からレジスタファイル 25 が最初に更新される迄の期間中、フォーク要求 7 a がスレッド管理部 3 に送出し続けられる。

【0115】スレッド管理部 3 は、プロセッサ 1-i からフォーク要求 7 a が送出されている期間内で、隣接プロセッサ 1-j に対するフォークが可能になると、プロセッサ 1-i にフォーク応答 7 b を送信すると同時に他プロセッサ 1-j に対してスレッド開始要求 7 c を送出することで子スレッドをフォークする（ステップ S 63）。プロセッサ 1-i に出されたフォーク応答 7 b によって、フォークドビット 28 がセットされると共にフォーク有効ビット 33 はリセットされ、またレジスタ転送ユニット 26 によってレジスタファイル 25 の内容がフォーク先プロセッサ 1-j に転送される（ステップ S 81）。なお、レジスタ転送ユニット 26 によるレジスタファイル 25 の転送中、実行ユニット 23 からのレジスタファイル 26 への書き込みが待たされる。

【0116】プロセッサ 1-i で命令の実行が進み、PC 21 の値がレジスタ 27 に保存されたフォーク先アドレスに一致したときの動作は第 1 の実施例と同様である。

【0117】本実施例のマルチスレッド実行方法の実行シーケンスの一例を図 28 に示す。フォーク命令 A から下に延びる矢印は、プロセッサ #0 においてレジスタファイル 25 が全く更新されていない期間を示す。この図 28 は、プロセッサ #0 で実行しているスレッドの最初のフォーク命令 A の時点で、フォーク先プロセッサ #1 がビジー状態であったが、プロセッサ #0 のレジスタファイル 25 が全く更新されていない期間内にフォーク先プロセッサ #1 がフリー状態になった場合を想定している。本実施例では、このような場合はフォーク命令 A によるフォークが行われる。

【0118】以上説明したように本実施例によれば、第 1 の実施例と同様な効果が得られると共に、親スレッドの実行開始後、フォーク命令の時点でフォークできなくても、レジスタファイルが更新される前に隣接プロセッサがフリー状態になればフォークを行うため、第 1 の実施例に比べてフォークされる可能性が高まり、スレッド実行の並列度が向上する。

【0119】

【第 2 の発明の第 3 の実施例】第 2 の実施例では、親スレッドのフォーク時点でフォーク可能でなければフォークを一旦保留にし、親スレッドのレジスタファイルが更新される前に子スレッドの実行を開始できる他プロセッサが生じなかった場合に当該フォークを断念したが、本実施例では、親スレッドのレジスタファイルが更新されても、その更新が子スレッドに継承すべきレジスタでなければフォークを行う。以下、第 2 の実施例との相違点を中心に本実施例を説明する。

【0120】図 29 を参照すると、本実施例における並列プロセッサシステムの各々のプロセッサ 1-i は、図 26 に示した構成に加えて、レジスタファイル 25 の各レジスタ 24-k（k=0~m）に 1 対 1 に対応し、対

応するレジスタ 24-k が子スレッドへ継承すべきレジスタであるときに限りセットされるクリエイティブット 36-k と、各レジスタ 24-k に 1 対 1 に対応し、対応するレジスタ 24-k のクリエイティブット 36-k の出力と実行ユニット 23 がレジスタ 24-k を更新したときに出力するレジスタ更新信号 37-k とを入力とするアンドゲート 38-k と、アンドゲート 38-k の出力の論理和信号であるフォーク無効信号 40 を出力するオアゲート 39 とを含んで構成されている。そして、図 26 のレジスタ更新信号 35 に代えて、フォーク無効信号 40 がフォーク有効ビット 33 にリセット信号として出力されている。また、各クリエイティブット 36-k の値がレジスタ転送ユニット 26 に出力されており、レジスタ転送ユニット 26 はレジスタファイル 25 のレジスタ 24-k のうち、対応するクリエイティブット 36-k がセットされているレジスタのみをフォーク先プロセッサのレジスタファイルに転送するように構成されている。

【0121】スレッドの開始から終了までのプロセッサ 1-i の処理の概要を図 30 に示す。スレッド管理部 3 からのスレッド開始要求 7c に基づき、プロセッサ 1-i で 1 つのスレッドの実行が開始される際、当該プロセッサ 1-i のフォークドビット 28 及びフォーク有効ビット 33 がリセットされる（ステップ S71）。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される（ステップ S72）。

【0122】実行ユニット 23 でデコードされた命令がフォーク命令の場合（ステップ S74 で YES）、実行ユニット 23 はフォークドビット 28 が 1 にセットされていれば（ステップ S75 で YES）、今回のフォーク命令を無効化し、次の命令を実行する（ステップ S72）。他方、フォークドビット 28 が 0 であれば（ステップ S75 で NO）、実行ユニット 23 はフォーク先アドレスをレジスタ 27 に保存すると共に、全てのクリエイティブット 36-k のセットアップを行う（ステップ S76）。つまり、レジスタファイル 25 のレジスタ 24-k のうち、子スレッドに継承すべきレジスタに対応するクリエイティブット 36-k はセットし、継承する必要のないレジスタに対応するクリエイティブット 36-k はリセットされたままにする。そして、フォーク信号 34 によってフォーク有効ビット 33 をセットすることにより（ステップ S77）、フォーク先アドレスを伴ったフォーク要求 7a をスレッド管理部 3 に送信する。また、実行ユニット 23 はレジスタファイル 25 中のレジスタ 24-k を更新すると（ステップ S91 で YES）、その更新したレジスタ 24-k に対応するレジスタ更新信号 37-k を論理 “1” とする（ステップ S92）。これにより、若し更新されたレジスタ 24-k が子スレッドへ継承すべきレジスタであった場合、そのレジスタ 24-k に対応するアンドゲート 38-k の出力が論理 “1” となり、オアゲート 39 からフォーク無効信号 4

0 が出力されてフォーク有効ビット 33 がリセットされる。つまり、プロセッサ 1-i からは、フォーク命令実行時点からレジスタファイル 25 中の子スレッドへの継承レジスタの何れかが最初に更新される迄の期間中、フォーク要求 7a がスレッド管理部 3 に送出し続けられる。

【0123】スレッド管理部 3 は、プロセッサ 1-i からフォーク要求 7a が送出されている期間内で、隣接プロセッサ 1-j に対するフォークが可能になると、プロセッサ 1-i にフォーク応答 7b を送信すると同時に他プロセッサ 1-j に対してスレッド開始要求 7c を送出することで子スレッドをフォークする（ステップ S63）。プロセッサ 1-i に出されたフォーク応答 7b によって、フォークドビット 28 がセットされると共にフォーク有効ビット 33 はリセットされ、またレジスタ転送ユニット 26 によってレジスタファイル 25 のレジスタの内、少なくとも子スレッドに継承すべきレジスタがフォーク先プロセッサ 1-j に転送される（ステップ S81）。

【0124】プロセッサ 1-i で命令の実行が進み、PC21 の値がレジスタ 27 に保存されたフォーク先アドレスに一致したときの動作は第 2 の実施例と同様である。

【0125】本実施例のマルチスレッド実行方法では、図 28 中のフォーク命令 A から下に述べる矢印は、プロセッサ #0 においてレジスタファイル 25 のレジスタの内、子スレッドに継承すべきレジスタが全く更新されていない期間となる。従って、第 1 の実施例は勿論のこと、第 2 の実施例に比べてもフォークの可能性をより高めることができる。

【0126】本実施例では、親スレッドのフォーク点で子スレッドへ継承すべきレジスタが判明している必要がある。このため、図 10 に示したコンパイラ 41 における制御及びデータフロー解析部 44 では、フォークする子スレッド毎に、親スレッドから子スレッドへ継承すべきレジスタを調査し、並列化コード挿入部 45 ではその調査結果に基づいて、子スレッドへ継承すべきレジスタを指定する記述を並列化プログラム 43 に挿入する。子スレッドへ継承すべきレジスタの指定は、フォーク命令で指定する方法、フォーク命令とは別の専用の命令で指定する方法などが利用できる。

【0127】なお、本実施例におけるレジスタ転送ユニット 26 は、クリエイティブット 36-k を参照することにより、親スレッドのレジスタファイル 25 のうち子スレッドに継承すべきレジスタだけをフォーク先プロセッサのレジスタファイルに転送するようにしたが、別の実施例として、レジスタファイル 25 の先頭のレジスタから順に所定の順番でレジスタの転送を行うシーケンスを開始し、クリエイティブット 36-k がセットされているレジスタの全ての転送が完了した時点で転送シーケンス

を停止するようにしても良い。この方法では、子スレッドに継承する必要のないレジスタも転送される場合があるが、転送シーケンスが簡素化される利点がある。勿論、別の実施例として、クリエイティブビット 36-k を一切参照せずに常に全レジスタを転送するようにレジスタ転送ユニット 26 が構成されていても良い。更に、子スレッドに継承すべきレジスタでも、フォーク先プロセッサの当該レジスタの値がフォーク時点で既に親スレッド側と同じ値に同じ値になっている場合にはあえて転送する必要がない点に着目して、子スレッドに継承すべきレジスタのうち、親スレッド側と異なる値になっているレジスタを検出し、この検出したレジスタだけをレジスタ転送ユニット 26 からフォーク先プロセッサに転送するようにしても良い。

【0128】以上説明したように本実施例によれば、第 2 の実施例と同様な効果が得られると共に、親スレッドのレジスタファイルの更新があっても、その更新が子スレッドに継承すべきレジスタでなければフォークを行うため、第 2 の実施例に比べてフォークされる可能性をより高めることができ、従ってスレッド実行の並列度をより向上することができる。

【0129】

【第 2 の発明の第 4 の実施例】本実施例は、親スレッドの実行開始後、フォーク命令の時点で隣接プロセッサがビジー状態であった場合、隣接プロセッサがフリー状態になるまでフォーク命令の実行をウェイトするようにした点で、第 1 乃至第 3 の実施例と相違する。以下、第 2 の実施例との相違点を中心に本実施例を説明する。

【0130】図 31 を参照すると、本実施例における並列プロセッサシステムの各々のプロセッサ 1-i は、図 26 に示した構成とほぼ同様な構成を有するが、スレッド管理部 3 から送信されるフォーク応答 7b が実行ユニット 23 にも入力されており、実行ユニット 23 はフォーク命令の実行時、フォーク応答 7b が返却されるまでフォーク命令の実行をウェイトする点で相違する。また、フォーク有効ビット 33 を実行ユニット 23 からリセットするレジスタ更新信号 35 は存在しない。

【0131】スレッドの開始から終了までのプロセッサ 1-i の処理の概要を図 32 に示す。スレッド管理部 3 からのスレッド開始要求 7c に基づき、プロセッサ 1-i で 1 つのスレッドの実行が開始される際、当該プロセッサ 1-i のフォークドビット 28 及びフォーク有効ビット 33 がリセットされる (ステップ S71)。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される (ステップ S72)。

【0132】実行ユニット 23 でデコードされた命令がフォーク命令の場合 (ステップ S74 で YES)、実行ユニット 23 はフォークドビット 28 が 1 にセットされていれば (ステップ S75 で YES)、今回のフォーク命令を無効化し、次の命令を実行する (ステップ S7

2)。他方、フォークドビット 28 が 0 であれば (ステップ S75 で NO)、フォーク先アドレスをレジスタ 27 に保存し (ステップ S76)、フォーク有効ビット 33 をセットすることにより、レジスタ 27 に保存したフォーク先アドレスを伴ったフォーク要求 7a をスレッド管理部 3 に送信する (ステップ S77)。そして、スレッド管理部 3 からフォーク応答 7b が返却されるのを待つ (ステップ S101)。

【0133】スレッド管理部 3 は、プロセッサ 1-i からフォーク要求 7a が送出されている期間内で、隣接プロセッサ 1-j に対するフォークが可能になると、プロセッサ 1-i にフォーク応答 7b を送信すると同時に他プロセッサ 1-j に対してスレッド開始要求 7c を送出することで子スレッドをフォークする (ステップ S63)。プロセッサ 1-i は、スレッド管理部 3 からフォーク応答 7b を受信すると、フォークドビット 28 をセットすると共にフォーク有効ビット 33 をリセットし、レジスタ転送ユニット 26 によってレジスタファイル 25 の内容をフォーク先プロセッサ 1-j に転送する (ステップ S102)。

【0134】プロセッサ 1-i で命令の実行が進み、PC21 の値がレジスタ 27 に保存されたフォーク先アドレスに一致したときの動作は第 2 の実施例と同様である。

【0135】

【第 2 の発明の第 5 の実施例】本実施例は、親スレッドの実行開始後、フォーク命令の時点で隣接プロセッサがビジー状態の場合、レジスタファイルの内容を退避させ、隣接プロセッサがフリー状態になった時点で前記退避した情報に基づいて子スレッドのフォークを行うようにした点で、第 1 乃至第 4 の実施例と相違する。以下、第 4 の実施例との相違点を中心に本実施例を説明する。

【0136】図 33 を参照すると、本実施例における並列プロセッサシステムの各々のプロセッサ 1-i は、図 31 に示した構成に加えて、退避バッファ 41 と、退避バッファ有効ビット 42 と、実行ユニット 23 から出力される退避信号 43 によって起動されるとレジスタファイル 25 の内容を退避バッファ 41 に退避する退避ユニット 44 とを備え、レジスタ転送ユニット 26 はレジスタファイル 25 及び退避バッファ 42 に接続されている。退避バッファ有効ビット 42 は、実行ユニット 23 から出力されるフォーク信号 34 によってリセットされ、退避信号 43 によってセットされ、また退避バッファ 41 に基づくフォークが行われた場合にリセットされる。

【0137】スレッドの開始から終了までのプロセッサ 1-i の処理の概要を図 34 に示す。スレッド管理部 3 からのスレッド開始要求 7c に基づき、プロセッサ 1-i で 1 つのスレッドの実行が開始される際、当該プロセッサ 1-i のフォークドビット 28 及びフォーク有効ビ

10

20

30

40

50

ット 33 がリセットされる (ステップ S71)。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される (ステップ S72)。

【0138】実行ユニット 23 でデコードされた命令がフォーク命令の場合 (ステップ S74 で YES)、実行ユニット 23 はフォークドビット 28 が 1 にセットされていれば (ステップ S75 で YES)、今回のフォーク命令を無効化し、次の命令を実行する (ステップ S72)。他方、フォークドビット 28 が 0 であれば (ステップ S75 で NO)、フォーク先アドレスをレジスタ 27 に保存し (ステップ S76)、フォーク有効ビット 33 をセットすることにより、レジスタ 27 に保存したフォーク先アドレスを伴ったフォーク要求 7a をスレッド管理部 3 に送信する (ステップ S77)。このとき退避バッファ有効ビット 42 がリセットされる。そして、スレッド管理部 3 から所定の時間内にフォーク応答 7b が返却された場合 (ステップ S111 で YES)、フォークドビット 28 をセットすると共にフォーク有効ビット 33 をリセットし、レジスタ転送ユニット 26 によってレジスタファイル 25 の内容をフォーク先プロセッサ 1-j に転送する (ステップ S112)。

【0139】他方、スレッド管理部 3 から所定の時間内にフォーク応答 7b が返却されなかった場合 (ステップ S111 で NO)、退避信号 43 によって退避ユニット 44 を起動することによりレジスタファイル 25 の内容を退避バッファ 41 へ退避させ、退避バッファ有効ビット 42 をセットする (ステップ S113)。この退避バッファ有効ビット 42 がセットされている間に、スレッド管理部 3 からフォーク応答 7b を受信すると (ステップ S114、S115 で YES)、レジスタ転送ユニット 26 により退避バッファ 41 に退避されているレジスタファイル 25 の内容をフォーク先プロセッサ 1-j に転送する (ステップ S116)。このとき、フォークドビット 28 がセットされ、フォーク有効ビット 33 及び退避バッファ有効ビット 42 がリセットされる。なお、次のフォーク命令が実行される迄に前回のフォーク命令に対するフォーク応答 7b がスレッド管理部 3 から返却されなかった場合、今回のフォーク命令の実行により退避バッファ有効ビット 42 はリセットされるので (ステップ S76)、前回のフォーク命令は結果的に無効化される。

【0140】プロセッサ 1-i で命令の実行が進み、PC21 の値がレジスタ 27 に保存されたフォーク先アドレスに一致したときの動作は第 4 の実施例と同様である。

【0141】

【第 2 の発明の第 6 の実施例】第 1 乃至第 5 の実施例では、各プロセッサ 1-i はプログラムカウンタの値が有効な子スレッドの開始アドレスと一致したときにスレッドの処理を終了したが、本実施例では、各プロセッサ

は、各フォーク命令に対応して並列化プログラム中のフォーク先アドレスの直前に挿入されているターム命令のうち、有効な子スレッドをフォークしたフォーク命令に対応するターム命令によってスレッドの処理を終了する。以下、第 1 の実施例との相違点を中心に本実施例を説明する。

【0142】図 35 を参照すると、本実施例における並列プロセッサシステムの各々のプロセッサ 1-i は、図 23 に示した構成のプロセッサにおける一致回路 29 及びアンドゲート 30 が省略され、スレッド ID を保持するレジスタ 45 が設けられている。このレジスタ 45 は、スレッド管理部 3 からのスレッド開始要求 7c に付随するスレッド ID が初期設定され、プロセッサ 1-i でフォーク命令が実行される毎に、そのフォーク命令で指定されたスレッド ID がセットされる。更に、スレッド管理部 3 へのフォーク要求 7a には、レジスタ 27 に保存されたフォーク先アドレスとレジスタ 45 に保存されたスレッド ID とが付随する。

【0143】スレッド管理部 3 は、図 22 のステップ S63 でフォーク先プロセッサ 1-j へスレッド開始要求 7c を送信する際、プロセッサ 1-i からのフォーク要求 7a に付随するフォーク先アドレス及びスレッド ID を一緒に送信する。

【0144】スレッドの開始から終了までのプロセッサ 1-i の処理の概要を図 36 に示す。スレッド管理部 3 からのスレッド開始要求 7c に基づき、プロセッサ 1-i で 1 つのスレッドの実行が開始される際、当該プロセッサ 1-i のフォークドビット 28 がリセットされ、スレッド開始要求 7c に付随するスレッド ID がレジスタ 45 にセットされる (ステップ S71)。以後、スレッドの命令のフェッチ、デコード、実行が継続して実行される (ステップ S72)。

【0145】実行ユニット 23 でデコードされた命令がフォーク命令の場合 (ステップ S74 で YES)、実行ユニット 23 はフォークドビット 28 が 1 にセットされていれば (ステップ S75 で YES)、今回のフォーク命令を無効化し、次の命令を実行する (ステップ S72)。他方、フォークドビット 28 が 0 であれば (ステップ S75 で NO)、実行ユニット 23 はフォーク先アドレスをレジスタ 27 に保存すると共にそのフォーク命令で指定されたスレッド ID をレジスタ 45 に保存 (上書き) し (ステップ S76)、レジスタ 27 に保存したフォーク先アドレス及びレジスタ 45 に保存したスレッド ID を伴ったフォーク要求 7a をスレッド管理部 3 に送信する (ステップ S77)。スレッド管理部 3 は、隣接プロセッサ 1-j に対してフォークが可能ならば、要求元のプロセッサ 1-i に対してフォーク応答 7b を返却し、隣接プロセッサ 1-j に対してはフォーク先アドレス及びスレッド ID を付加したスレッド開始要求 7c を送出する。フォーク応答 7b を受信したプロセッサ 1

ー i は、フォークドビット 28 を 1 にセットし、レジスタ転送ユニット 26 によって親スレッドのレジスタファイル 25 の内容を通信バス 6 経由でフォーク先プロセッサ 1-j のレジスタファイルに転送するレジスタ継承操作を行う（ステップ S 81）。また、フォーク先プロセッサ 1-j では図 36 のステップ S 71 以降の処理を実行する。

【0146】他方、スレッド管理部 3 は、隣接プロセッサ 1-j に対してフォークが不可能ならば、プロセッサ 1-i から送信されたフォーク要求 7a を廃棄する。従って、プロセッサ 1-i で実行された今回のフォーク命令は無効化され、当該フォーク命令による子スレッドのフォークは断念される。

【0147】実行ユニット 23 でデコードされた命令がターム命令の場合（ステップ S 121 で YES）、実行ユニット 23 は、フォークドビット 28 が 1 にセットされており且つ当該ターム命令で指定されたスレッド ID がレジスタ 45 に保存されているスレッド ID と一致した場合（ステップ S 122、S 123 で YES）、当該ターム命令を実行することにより、スレッド終了通知 7d を送出し（ステップ S 79）、スレッド管理部 3 からスレッド終了許可 7e を受信した時点でスレッドの処理を終了する（ステップ S 80）。しかし、フォークドビット 28 が 1 にセットされていないか、ターム命令で指定されたスレッド ID がレジスタ 45 に保存されているスレッド ID と一致しない場合は、当該ターム命令を無効にし、PC 21 に従って命令の実行を継続して実行する（ステップ S 72）。

【0148】本実施例では、各プロセッサはターム命令によってスレッドの処理を終了するため、図 10 のコンパイラ 41 における並列化コード挿入部 45 は、子スレッドの開始点の直前に、当該子スレッドをフォークするフォーク命令に付加したスレッド ID と同じスレッド ID を持つターム命令を挿入する。

【0149】本実施例と同様に、第 2 乃至第 5 の実施例において、各プロセッサが各フォーク命令に対応して並列化プログラム中のフォーク先アドレスの直前に挿入されているターム命令のうち、有効な子スレッドをフォークしたフォーク命令に対応するターム命令によってスレッドの処理を終了するように変形することができる。

【0150】以上説明したように第 2 の発明によれば、フォーク 1 回制限の保証のない並列化プログラムであってもフォーク 1 回モデルによるマルチスレッド実行が可能になると共に、親スレッドの実行を開始したプロセッサで最初に子スレッドのフォークに成功したフォーク命令以外の全てのフォーク命令を無効化することで、プログラム実行時にフォーク 1 回制限を保証する為、動作が決定的であり、制御が簡単である利点がある。

【0151】以上、本発明を幾つかの実施例を挙げて説明したが、本発明は以上の実施例にのみ限定されず、そ

の他各種の付加変更が可能である。例えば、前記各実施例では、複数のプロセッサに共通にスレッド管理部 3 を設ける集中スレッド管理型の並列プロセッサシステムに本発明を適用したが、文献 1 等に記載されるように各プロセッサ毎にスレッド管理部を設ける分散スレッド管理型の並列プロセッサシステムにも本発明は適用可能である。また、プロセッサ相互間を通信バス 6 によって接続したが、リング型フォークモデルにあっては隣接するプロセッサ間どうしをリング上に通信線で接続する形態の並列プロセッサシステムに対しても本発明は適用可能である。

【0152】

【発明の効果】以上説明したように本発明によれば、フォーク 1 回制限の保証のない並列化プログラムであってもプログラム実行時にフォーク 1 回制限を保証できるマルチスレッド実行方法及び並列プロセッサシステムが得られ、コンパイル段階でのフォーク 1 回制限を取り除くことができる。

【0153】また、第 1 及び第 2 の発明ともプログラム実行時にフォーク 1 回制限を保証する動作が決定的であり、制御が簡単である効果がある。

【0154】また、第 2 の発明によれば、スレッド破棄命令（abort）無しで、制御依存投機処理が可能になる効果がある。

【0155】また、第 1 及び第 2 の発明における第 1 乃至第 3 の実施例によれば、退避バッファを持つことによるハードウェア量の増加、OS のプロセス切り替え時におけるオーバヘッドの増大を防止しつつ、親スレッドのフォーク命令時点で子スレッドを生成できる空きのプロセッサが存在しない場合でも処理の中断無しにプログラムの処理を支障なく遂行することができる効果がある。

【0156】また、第 1 及び第 2 の発明における第 2 の実施例によれば、フォーク命令の時点でフォークできなくても、レジスタファイルが更新される前に子スレッドの実行を開始できる空きのプロセッサが生じるとフォークが可能になるため、第 1 の実施例に比べてフォークできる確率が高まり、スレッド実行の並列度を向上することができる。

【0157】また、第 1 及び第 2 の発明における第 3 の実施例によれば、親スレッドのレジスタファイルの更新があっても、その更新が子スレッドに継承すべきレジスタでなければフォークを行うため、第 2 の実施例に比べてフォークできる確率を高めることができ、スレッド実行の並列度をより一層向上することができる。

【0158】また、第 1 及び第 2 の発明における第 1 乃至第 5 の実施例によれば、子スレッドの開始点の直前にターム命令を置く必要がなくなり、ターム命令の削減によってプログラムサイズをコンパクト化でき、命令メモリに必要な容量の削減、命令フェッチ数の削減による処理性能の向上が可能となる。

【0159】また、第1及び第2の発明における第6の実施例によれば、従来と同様に子スレッドの開始点の直前にターム命令が置かれたプログラムを支障なく実行することが可能となる。

【図面の簡単な説明】

【図1】本発明の作用の説明図である。

【図2】本発明の並列プロセッサシステムの一例を示すブロック図である。

【図3】本発明の並列プロセッサシステムにおけるスレッド管理部の構成例を示すブロック図である。

【図4】本発明の並列プロセッサシステムにおけるスレッド管理部のスレッド管理シーケンサがプロセッサからフォーク要求を受信した際の処理例を示すフローチャートである。

【図5】本発明の並列プロセッサシステムにおけるスレッド管理部のスレッド管理シーケンサがプロセッサからスレッド終了通知を受信した際の処理例を示すフローチャートである。

【図6】本発明の第1の発明の第1の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図7】本発明の第1の発明の第1の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図8】本発明の第1の発明の第1の実施例にかかるマルチスレッド実行方法の実行シーケンスの一例を示す図である。

【図9】本発明の第1の発明の第1の実施例にかかるマルチスレッド実行方法の実行シーケンスの別の例を示す図である。

【図10】本発明のマルチスレッド実行方法向けの並列化プログラムを生成するコンパイラの構成例を示すブロック図である。

【図11】本発明の第1の発明の第2の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図12】本発明の第1の発明の第2の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図13】本発明の第1の発明の第2の実施例にかかるマルチスレッド実行方法の実行シーケンスの一例を示す図である。

【図14】本発明の第1の発明の第3の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図15】本発明の第1の発明の第3の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャート

である。

【図16】本発明の第1の発明の第4の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図17】本発明の第1の発明の第4の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図18】本発明の第1の発明の第5の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図19】本発明の第1の発明の第5の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図20】本発明の第1の発明の第6の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図21】本発明の第1の発明の第6の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図22】本発明の並列プロセッサシステムにおけるスレッド管理部のスレッド管理シーケンサがプロセッサからフォーク要求を受信した際の別の実施例を示すフローチャートである。

【図23】本発明の第2の発明の第1の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図24】本発明の第2の発明の第1の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図25】本発明の第2の発明の第1の実施例にかかるマルチスレッド実行方法の実行シーケンスの一例を示す図である。

【図26】本発明の第2の発明の第2の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図27】本発明の第2の発明の第2の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図28】本発明の第2の発明の第2の実施例にかかるマルチスレッド実行方法の実行シーケンスの一例を示す図である。

【図29】本発明の第2の発明の第3の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図30】本発明の第2の発明の第3の実施例にかかる

並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図 3 1】本発明の第 2 の発明の第 4 の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

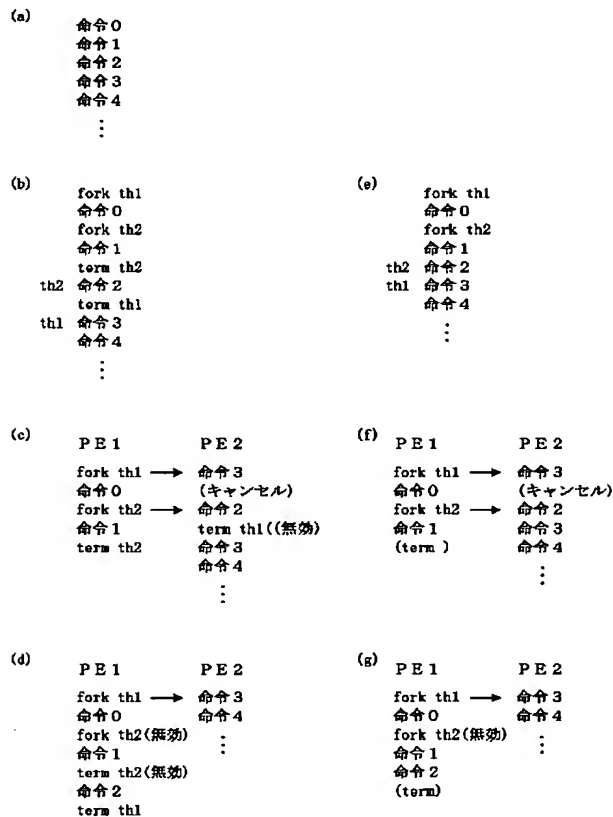
【図 3 2】本発明の第 2 の発明の第 4 の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図 3 3】本発明の第 2 の発明の第 5 の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図 3 4】本発明の第 2 の発明の第 5 の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図 1】

【図 1】



【図 3 5】本発明の第 2 の発明の第 6 の実施例にかかる並列プロセッサシステムにおけるプロセッサの構成例を示すブロック図である。

【図 3 6】本発明の第 2 の発明の第 6 の実施例にかかる並列プロセッサシステムにおけるスレッドの開始から終了までのプロセッサの処理の一例を示すフローチャートである。

【図 3 7】従来のマルチスレッド実行方法の処理の概要を示す図である。

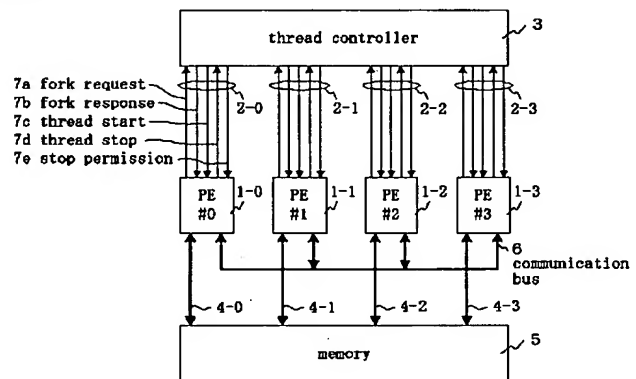
10 【図 3 8】従来の問題点の説明図である。

【符号の説明】

- 1-0~1-3…プロセッサ
- 2-0~2-3…信号線
- 3…スレッド管理部
- 4-0~4-3…信号線
- 5…メモリ
- 6…通信バス

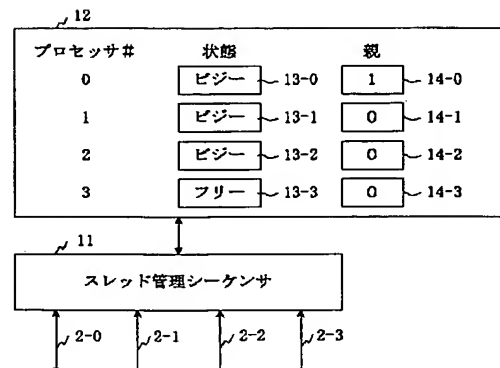
【図 2】

【図 2】

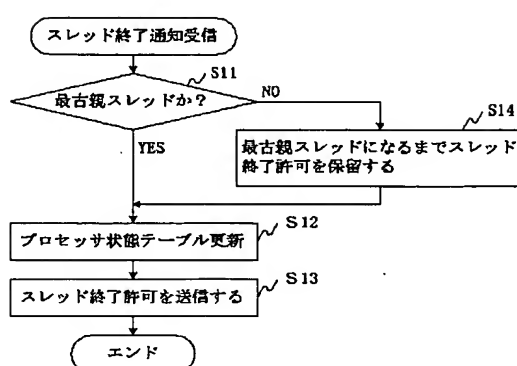
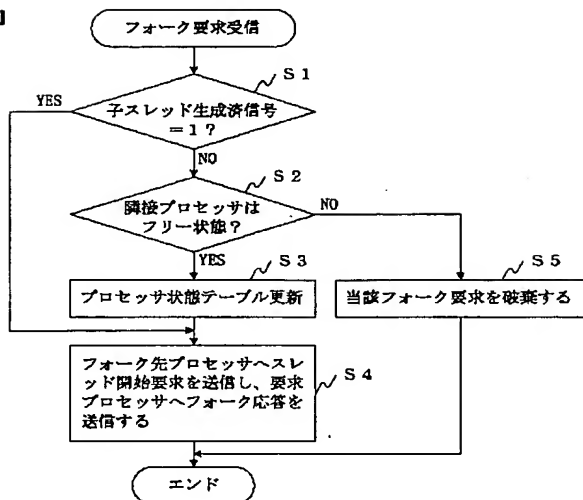


【図 3】

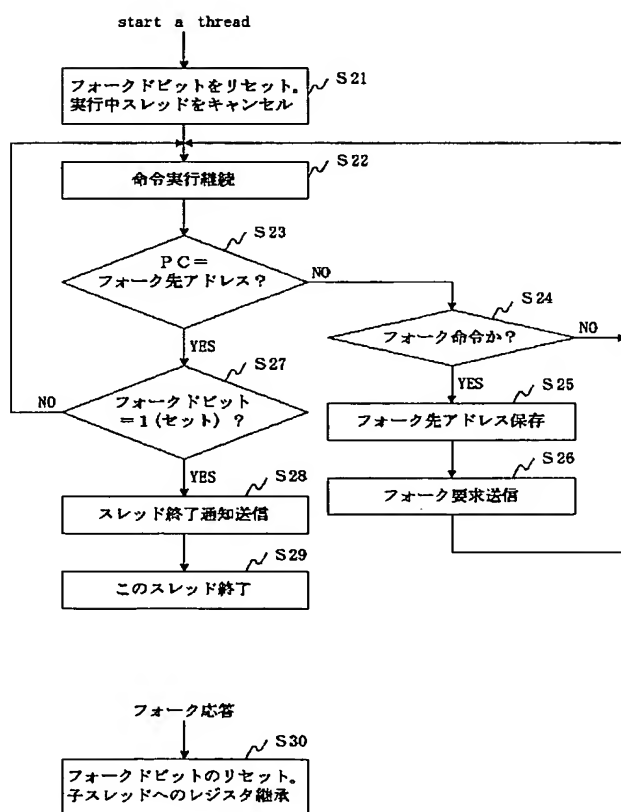
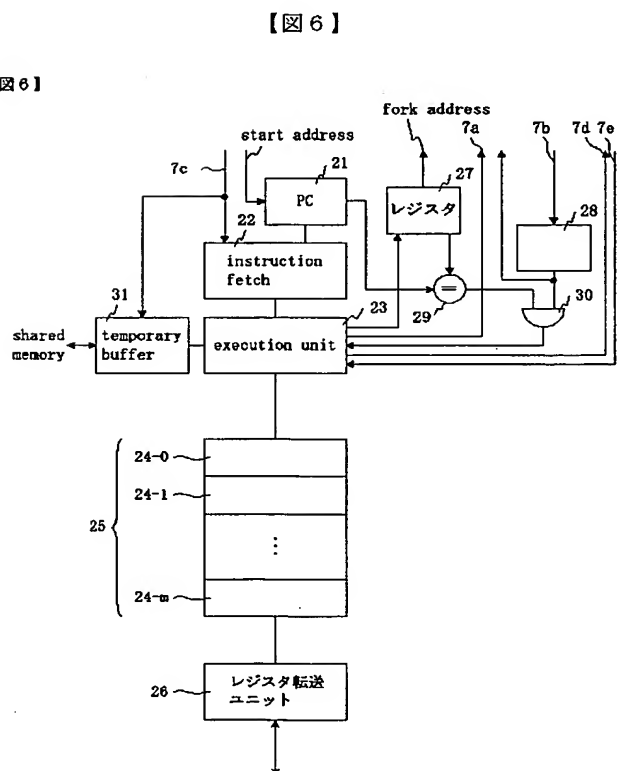
【図 3】



【图 5】

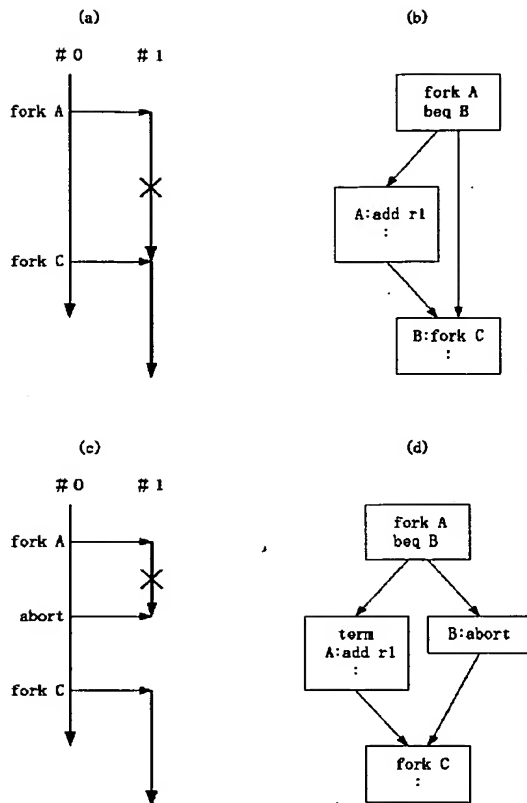


【图7】



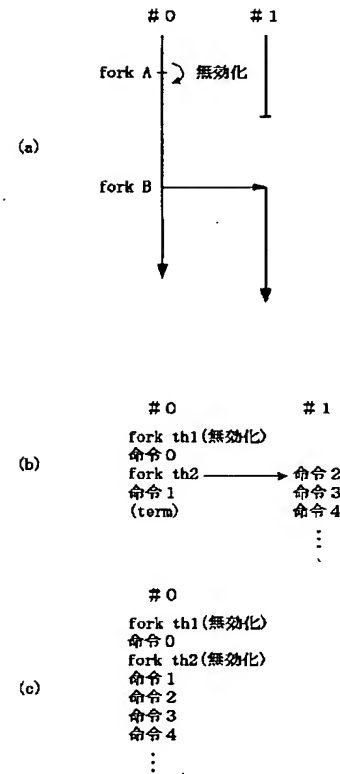
【図 8】

【図 8】



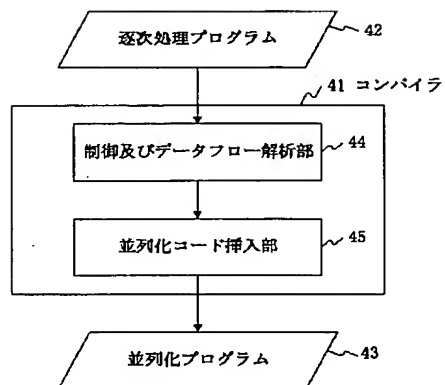
【図 9】

【図 9】



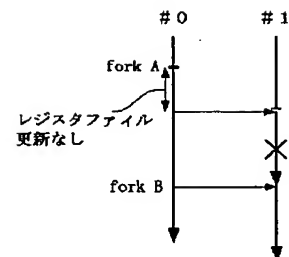
【図 10】

【図 10】



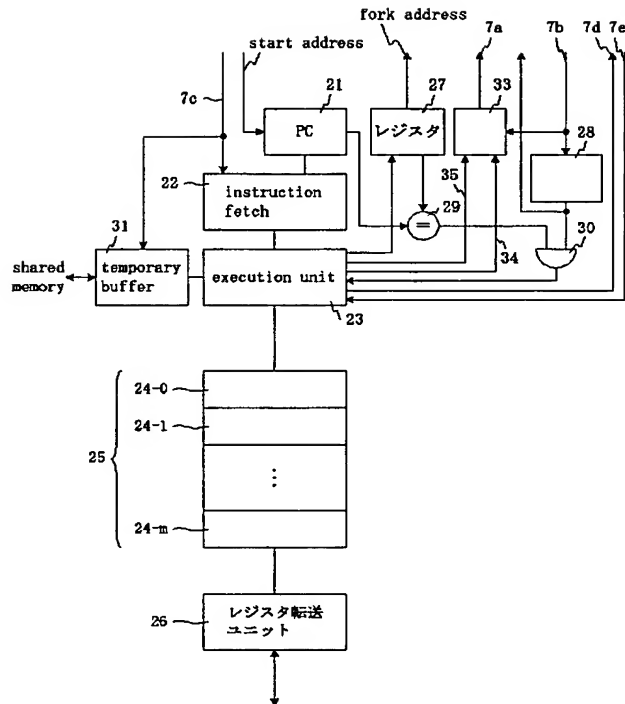
【図 13】

【図 13】



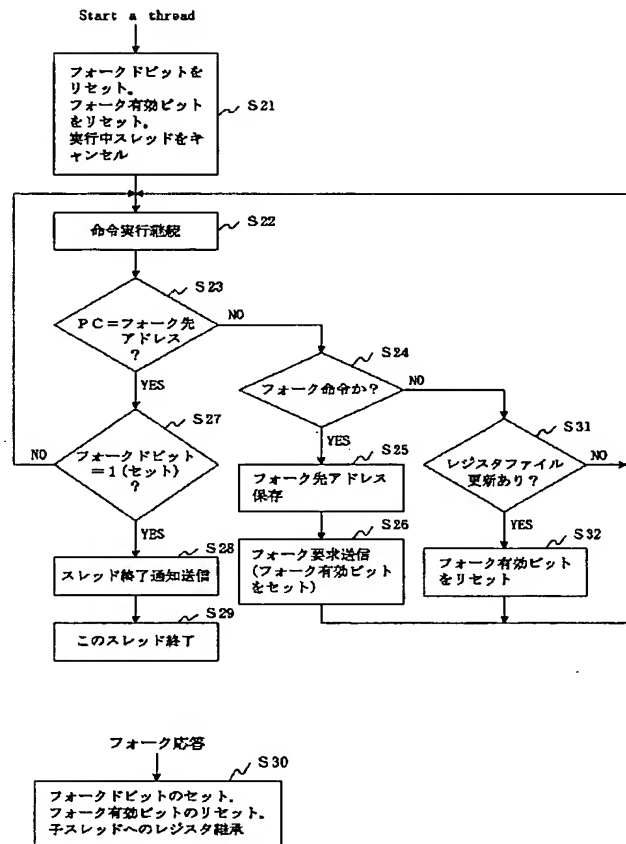
【図 11】

【図 11】



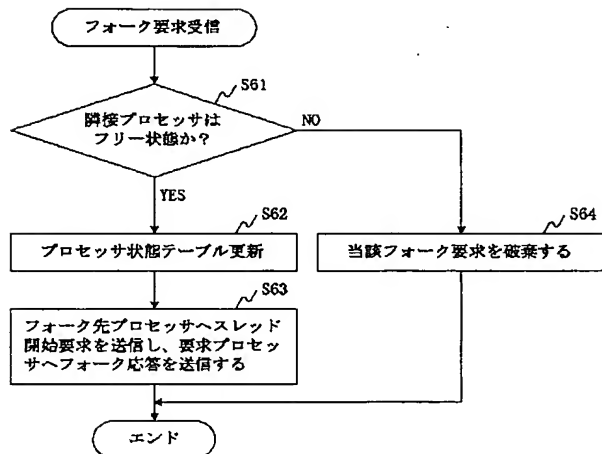
【図 12】

【図 12】



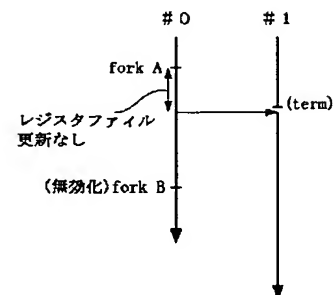
【図 22】

【図 22】



【図 28】

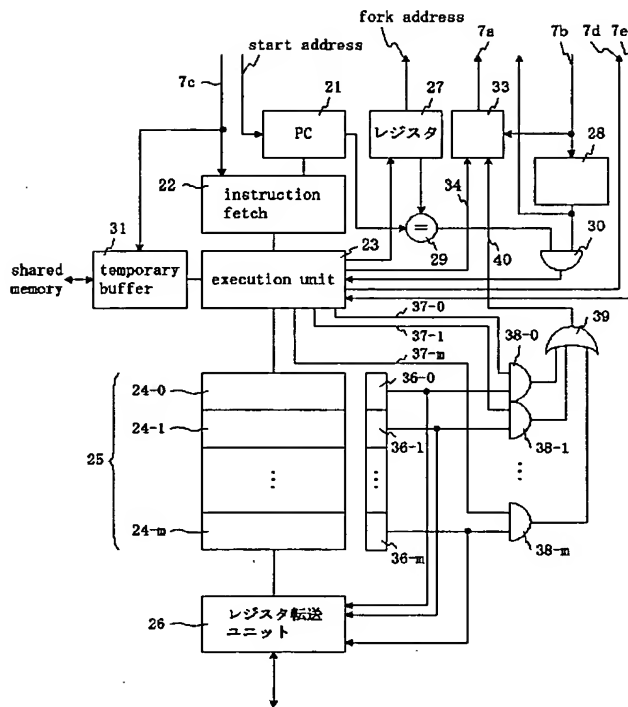
【図 28】



【図 14】

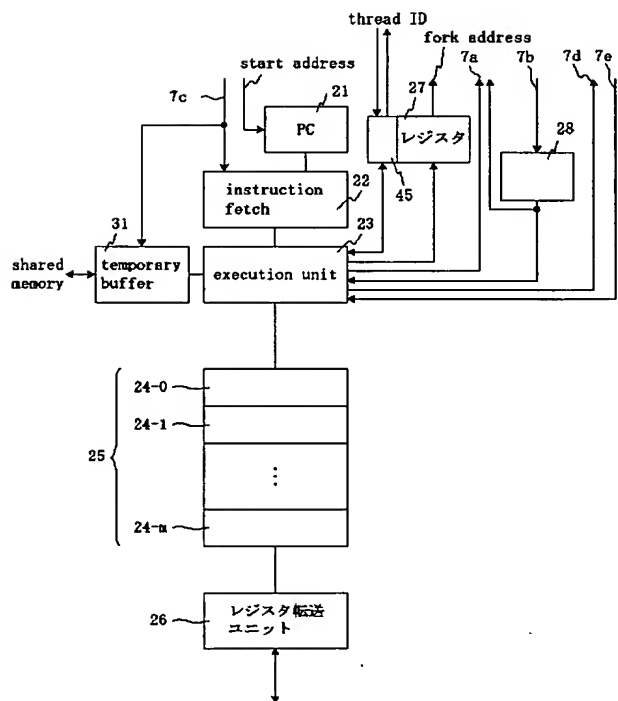
【図 15】

【図 14】

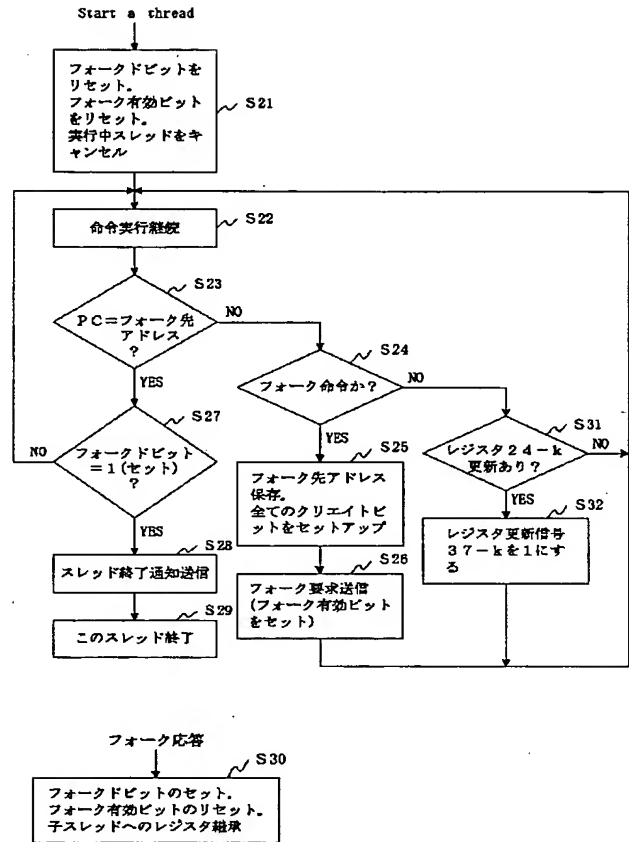


【図 20】

【図 20】

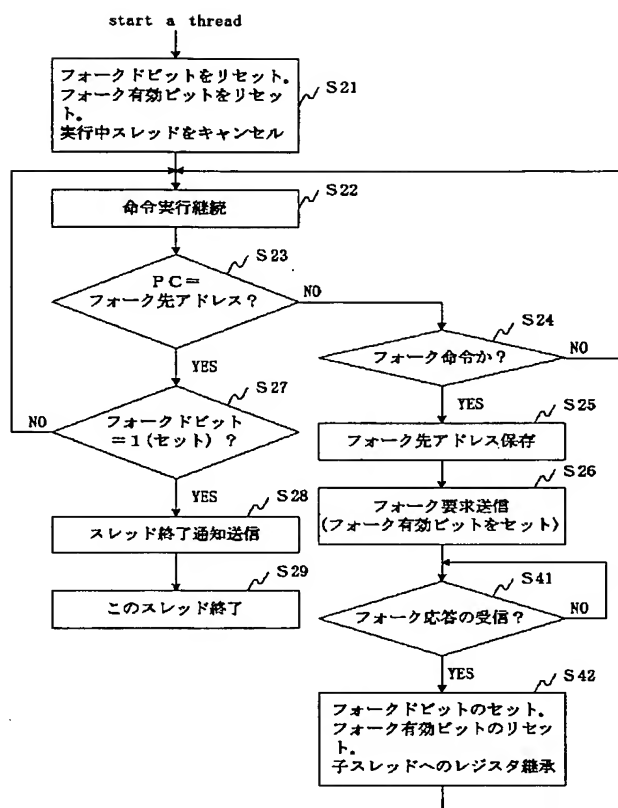


【図 15】

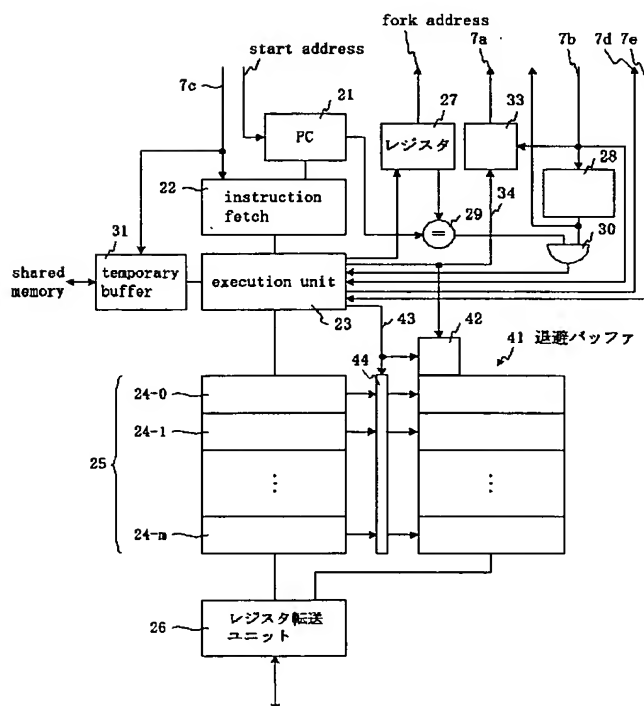


【图 17】

【图 17】

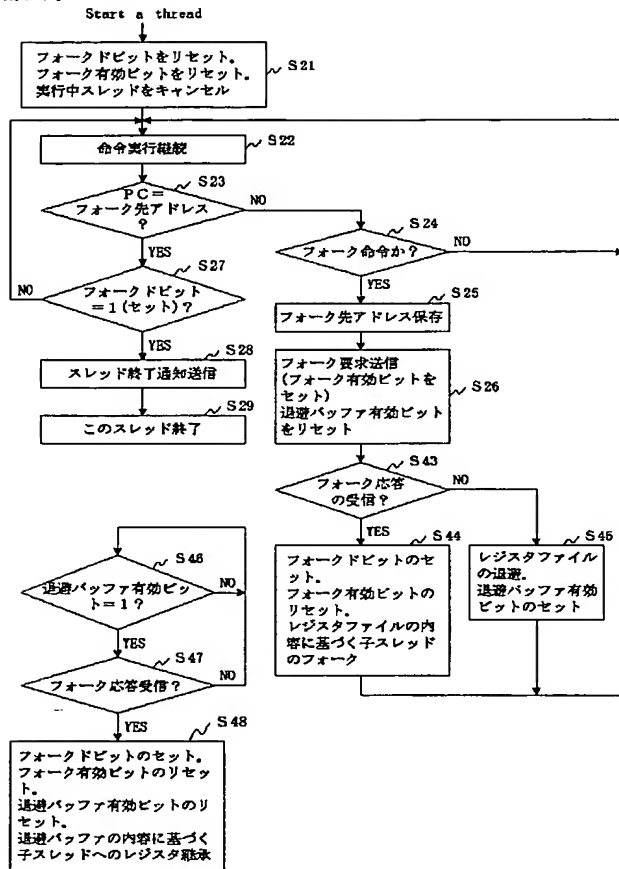


【图 18】



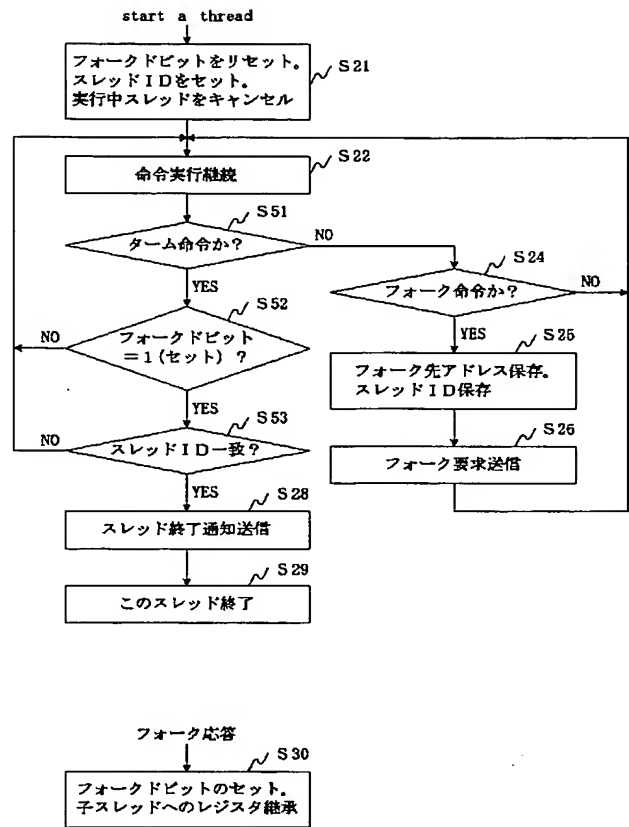
【図 19】

【図 19】



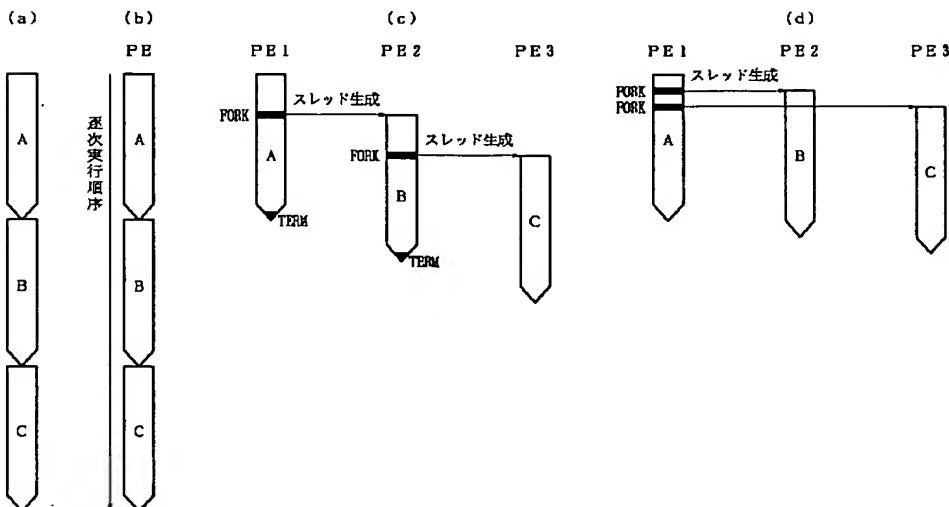
【図 21】

【図 21】



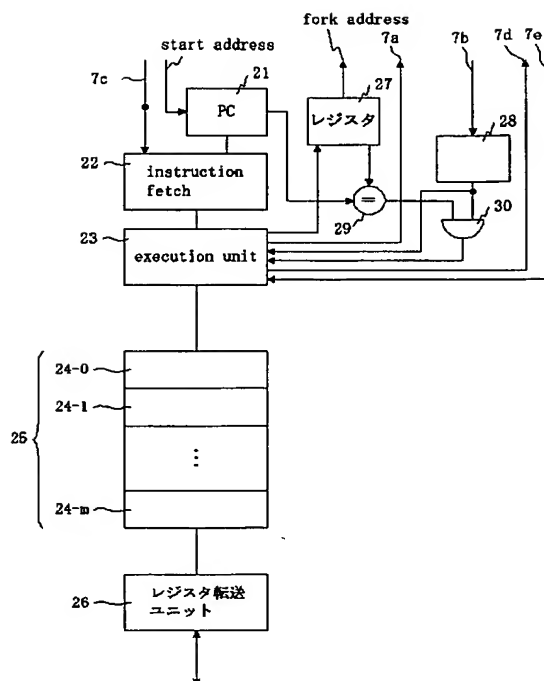
【図 3.7】

【図 3.7】



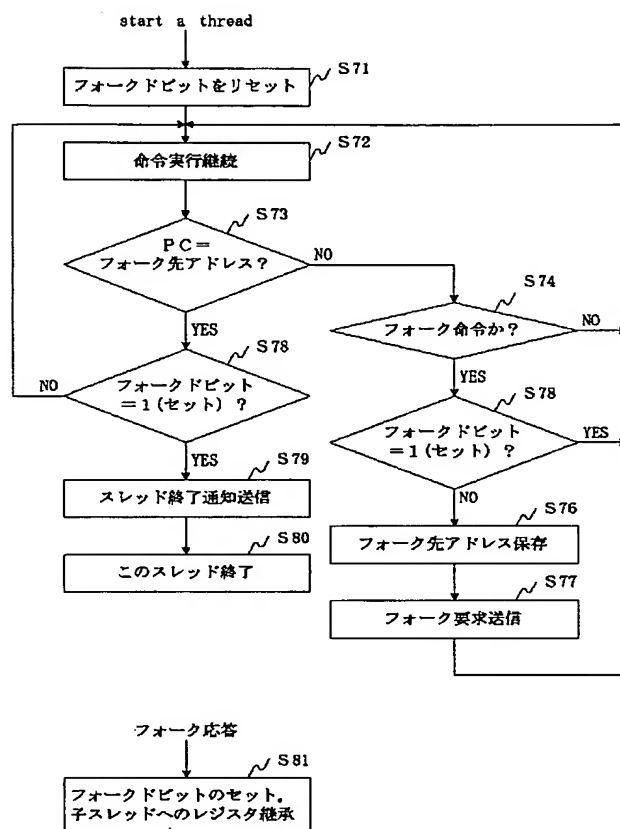
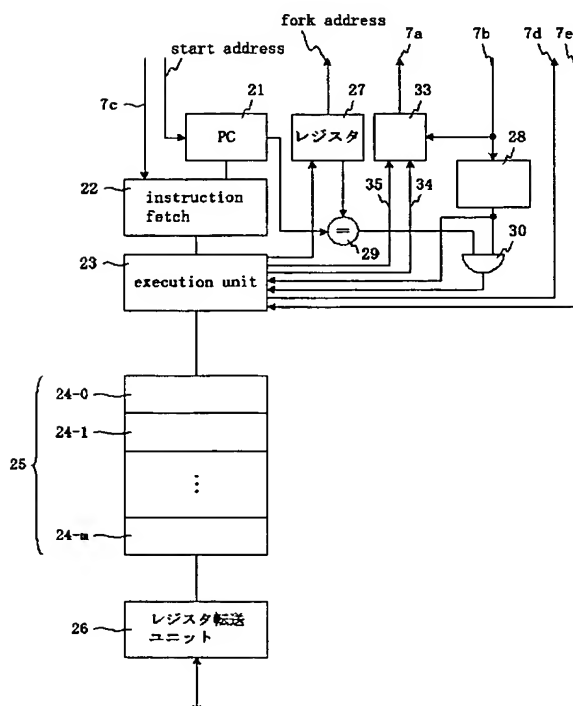
【图 24】

【图 24】



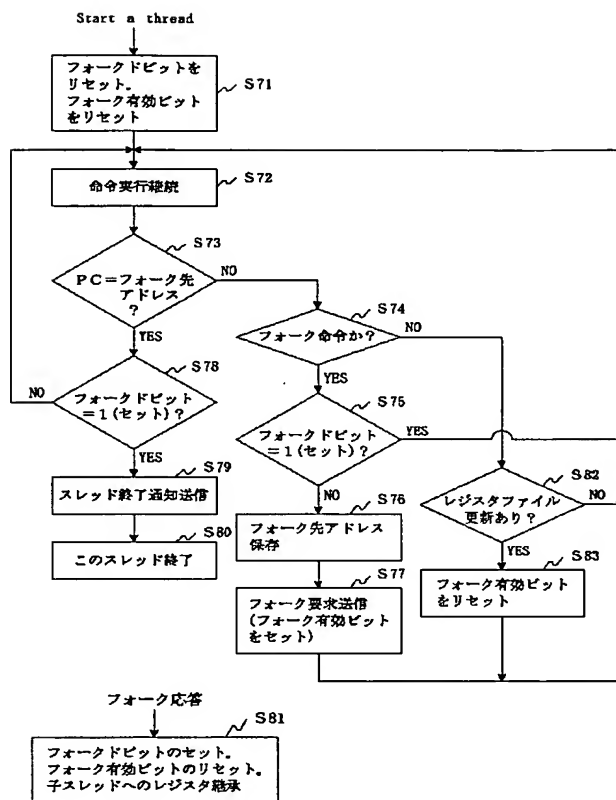
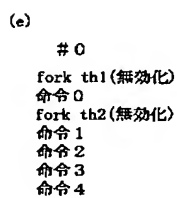
【图 26】

【图 26】



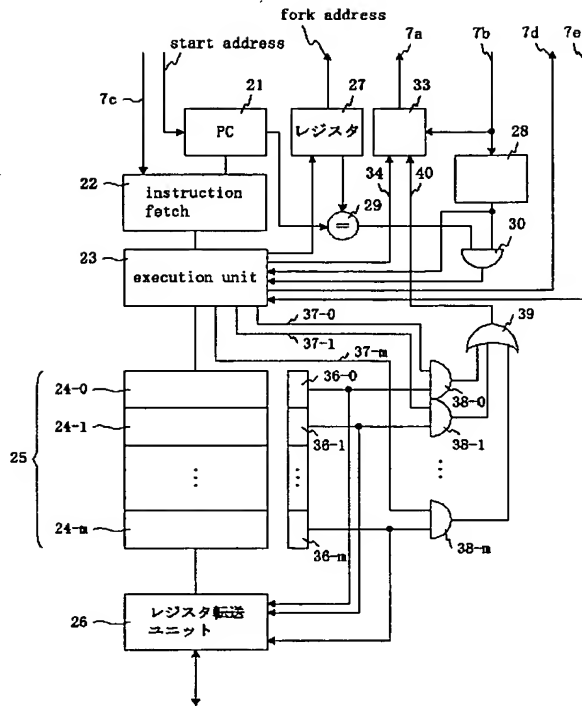
【图 27】

【圖 27】



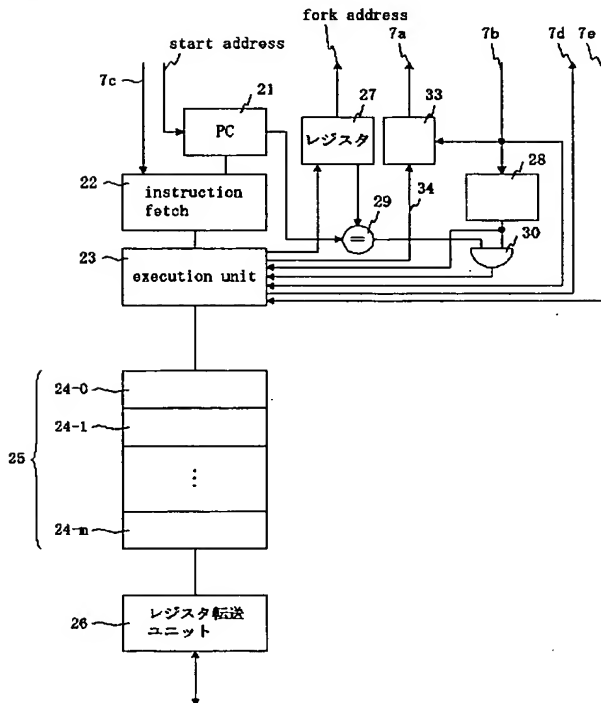
【図29】

【図29】



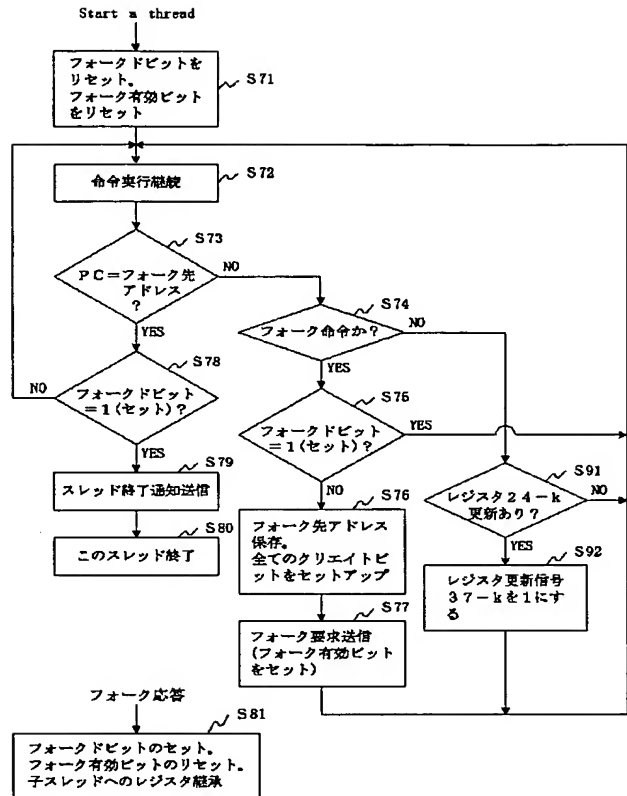
【図31】

【図31】



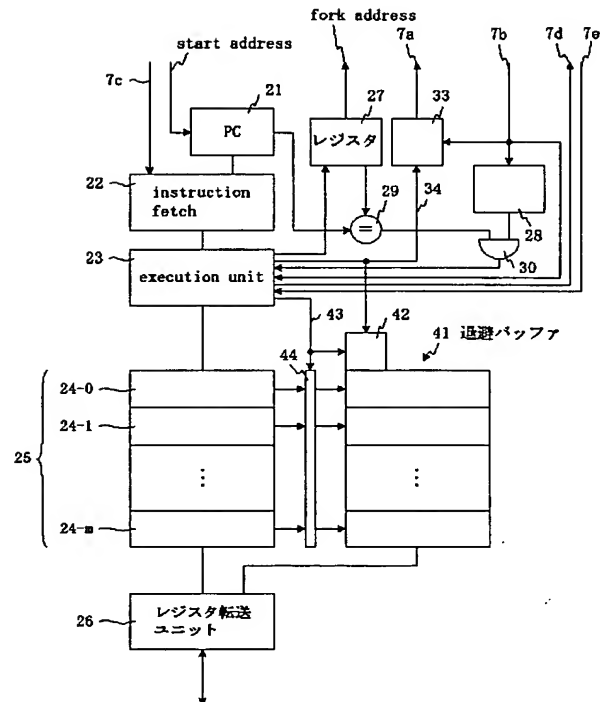
【図30】

【図30】



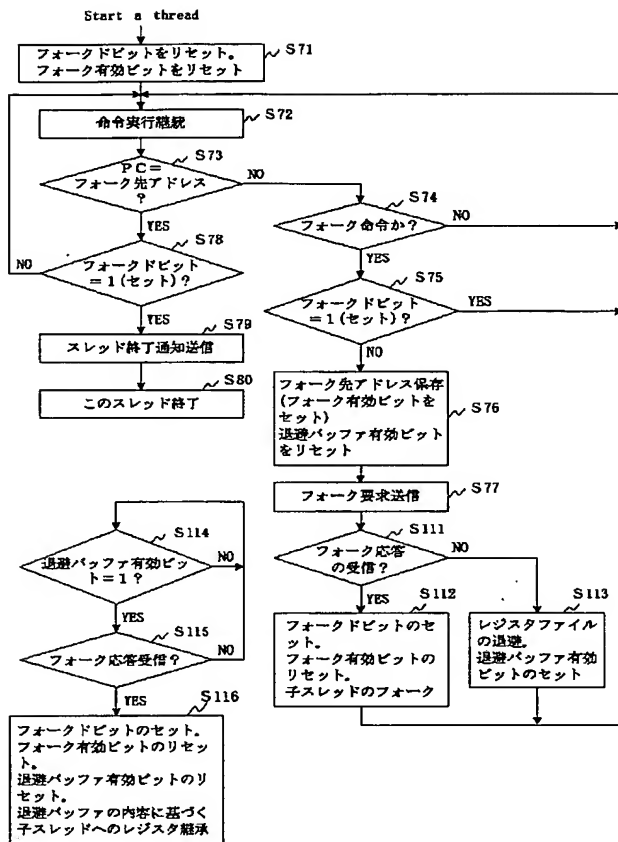
【图 3 3】

【图 3 3】



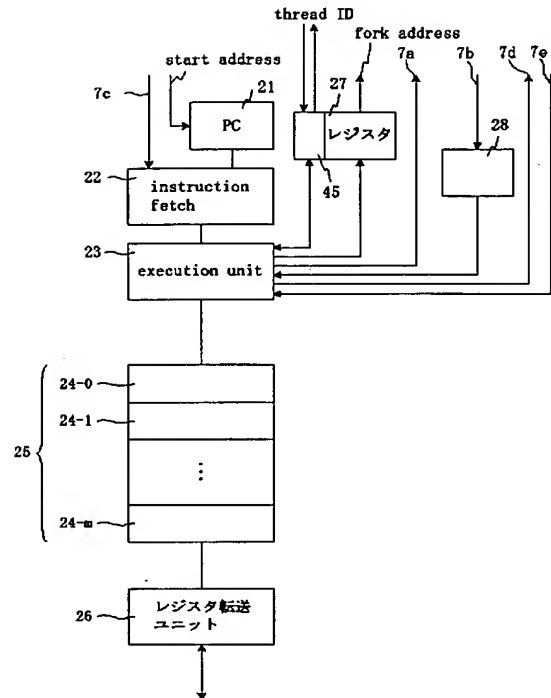
【図 34】

【図 34】



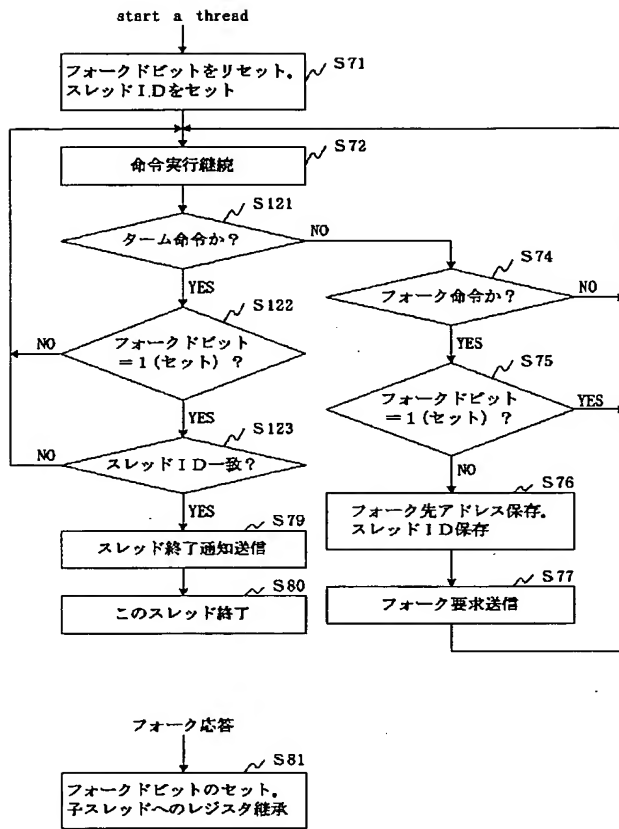
【図 35】

【図 35】



【図 36】

【図 36】



【図 38】

【図 38】

